DeadFace CTF 2023

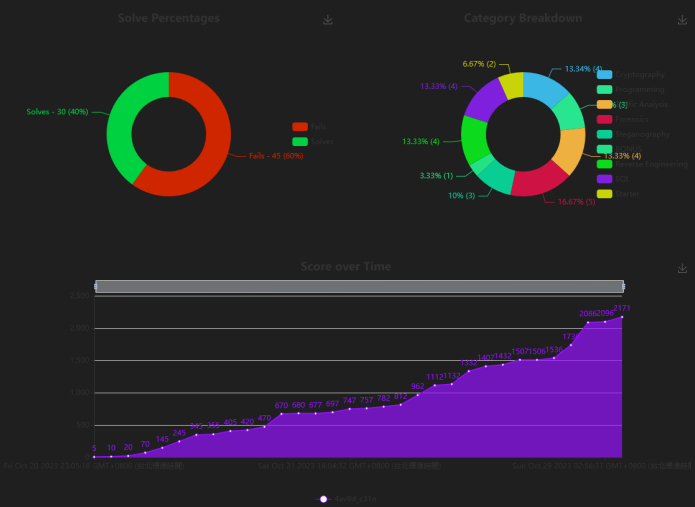
4av9d_c31n

165TH PLACE

2171 POINTS

Members

User Name	Score
4av9d_c31n Captain	670
sbk6401	1501
ccccc	0



Awards

Hint 4	Hint 7
hints	hints
Hint for Dead Drop	Hint for Fetching Secrets
-1	-3

Solves

Challenge	Category	Value	Time
Refill on Soup	Cryptography	75	October 29th, 3:27:22 AM
Letter Soup	Cryptography	10	October 29th, 2:56:31 AM
The CDR of the CAR... RAH, RAH, RAH!!	Programming	350	October 29th, 12:41:22 AM
Chatty Cathy	Programming	200	October 29th, 12:25:12 AM
Dead Drop	Programming	30	October 28th, 11:47:44 PM
UVB-76 (Hello, are you there?)	Traffic Analysis	75	October 28th, 10:11:45 PM
Sometimes IT Lets You Down	Traffic Analysis	25	October 28th, 6:12:12 PM
Malum	Forensics	75	October 28th, 5:20:36 PM
Host Busters 2	Forensics	200	October 28th, 4:50:43 PM
Fetching Secrets	Steganography	20	October 28th, 4:25:29 PM
Electric Steel	Steganography	150	October 28th, 4:03:13 PM
Tin Balloon	Forensics	150	October 28th, 3:51:29 PM
Off the Rails	BONUS	30	October 24th, 2:23:21 AM
B1Tz and B0tZ	Cryptography	25	October 24th, 12:41:24 AM
Coin Code	Cryptography	10	October 24th, 12:08:42 AM
Host Busters 1	Forensics	50	October 21st, 2:04:32 PM
What's the Wallet	Forensics	20	October 21st, 1:56:03 PM
You've Been Ransomwared	Steganography	10	October 21st, 1:09:28 PM
Cereal Killer 04	Reverse Engineering	200	October 21st, 8:44:58 AM
Starypax	SQL	50	October 21st, 5:10:37 AM
Credit Compromise	SQL	15	October 21st, 4:44:58 AM
Git Rekt	Traffic Analysis	50	October 21st, 4:41:52 AM
Cereal Killer 05	Reverse Engineering	10	October 21st, 4:35:52 AM
Cereal Killer 02	Reverse Engineering	100	October 21st, 4:11:03 AM
City Hoard	SQL	100	October 21st, 3:19:23 AM
Creepy Crawling	Traffic Analysis	75	October 21st, 2:12:29 AM
Cereal Killer 01	Reverse Engineering	50	October 21st, 1:46:22 AM
Aurora Compromise	SQL	10	October 20th, 11:35:44 PM
Starter 2	Starter	5	October 20th, 11:17:17 PM
Starter 1	Starter	5	October 20th, 11:05:18 PM

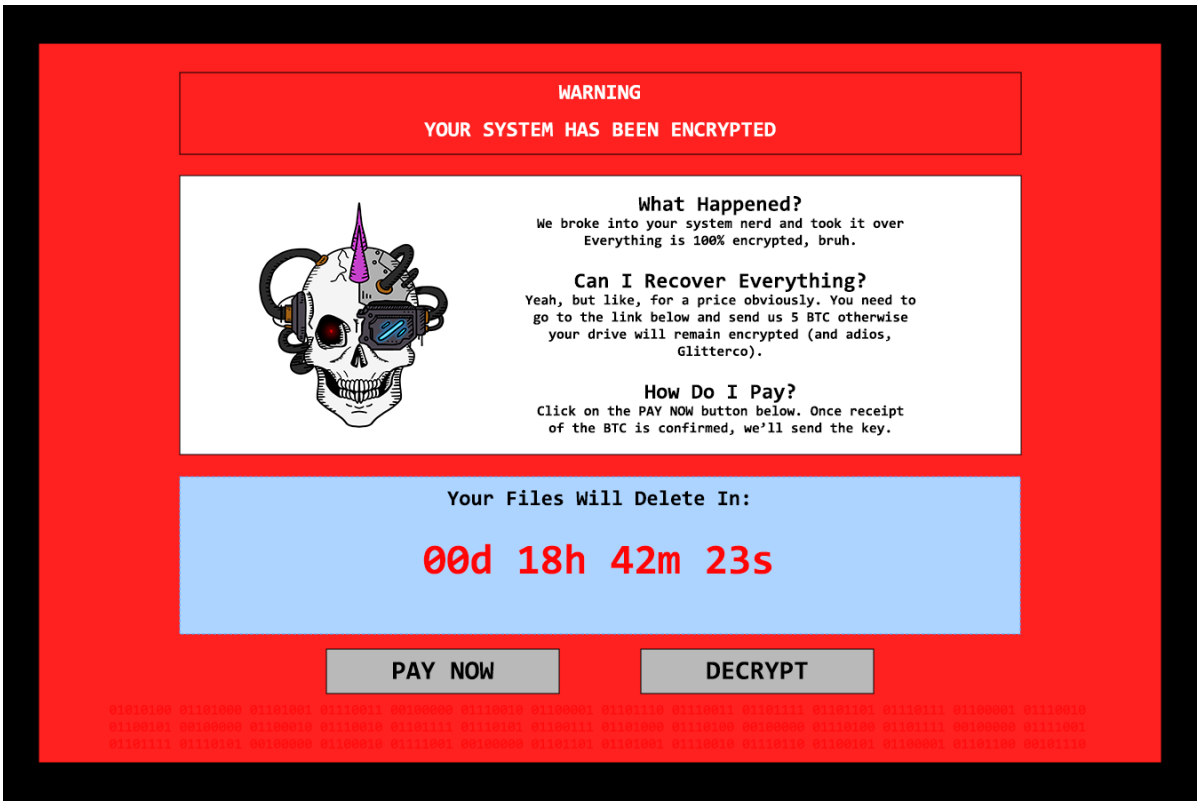
Steganography

You've Been Ransomwared

Description

> DEADFACE is taunting GlitterCo with their latest ransomware attack. According to our intel, the attackers like to leave a calling card in their attacks. If we can figure out which DEADFACE actor executed this attack, we might be able to figure out a way around paying. Can you find anything in this screenshot that might point to which attacker ran this ransomware attack?
>
> Submit the flag as flag{attacker_name}.

Attached File



Recon

最簡單的水題，用Stegsolve就可以了

Exploit

1. Using Stegsolve and Extract the Image

2. Convert to Char

```
>>> a = ['01010100', '01101000', '01101001', '01110011', '00100000',
'01110010', '01100001', '01101110', '01110011', '01101111', '01101101',
'01110111', '01100001', '01110010', '01100101', '00100000', '01100010',
'01110010', '01101111', '01110101', '01100111', '01101000', '01110100',
'00100000', '01110100', '01101111', '00100000', '01111001', '01101111',
'01110101', '00100000', '01100010', '01111001', '00100000', '01101101',
'01101001', '01110010', '01110110', '01100101', '01100001', '01101100',
'00101110']
>>> for byte in a:
...     print(bytes.fromhex(hex(int(byte, 2))[2:]).decode('utf-8'), end="")
...
This ransThis ransomware brought to you by mirveal.
```

Flag: `flag{mirveal}`

Fetching Secrets

Description

> This image was found on Ghost Town. Looks like one of DEADFACE's newest members is new to steganography. See if you can find any hidden information in this image. Knowing information about the image may help to reveal the flag.
>
> Submit the flag as: flag{flag_text}.

Recon

又學到一個新工具了-stegseek，然後這一題是參考 Steganography Pro TOOL : "Fetching Secrets" Walkthrough :ctf.deadface.io CTF 這一部影片，雖然我聽不懂看操作還是可以有樣學樣，主要就是利用這個工具爆破出steghide的密碼(真香.jpg)

Exploit - stegseek

```
$ stegseek ./Fetching\ Secrets.jpeg rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "kira"B)
[i] Extracting to "Fetching Secrets.jpeg.out".
```

flag{g00d_dawg_woofw00f}

Flag: `flag{g00d_dawg_woofw00f}`

Electric Steel

Description

> Check out this image DEADFACE left on one of their victims' machines. We tried a couple tools and they didn't reveal anything. Take a look and see what you can find.
>
> Submit the flag as flag{flag_text}.

Recon

這一題學到新的東西，應該說小地方沒有注意到，按照基本的recon技巧都做了(包含binwalk/pngcheck/exiftool/strings/file/stat...)，但都沒有甚麼發現，應該說其實會有不一樣的地方，這次是參考這一篇WP，然後發現到原來裡面有藏東西，可以用tar直接解壓縮拿到flag

:::info
binwalk -e代表extract；
tar -x代表extract；-f代表解壓的file
:::

Exploit

```
$ binwalk -e electric-steel.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1232 x 928, 8-bit/color RGB, non-
interlaced
2767          0xACF           Zlib compressed data, default compression
1435378       0x15E6F2        TIFF image data, big-endian, offset of first image
directory: 8
1435914       0x15E90A        Copyright string: "Copyright (c) 1998 Hewlett-
Packard Company"
```

```
1467642       0x1664FA        gzip compressed data, from Unix, last modified:
2023-06-04 01:14:27
$ cd _electric-steel.png-0.extracted
$ tar -xf 1664FA
$ ll
total 1444
drwxrwxrwx 1 sbk6401 sbk6401    4096 Oct 28 16:03 .
drwxrwxrwx 1 sbk6401 sbk6401    4096 Oct 28 16:02 ..
-rwxrwxrwx 1 sbk6401 sbk6401   10240 Oct 28 16:02 1664FA
-rwxrwxrwx 1 sbk6401 sbk6401       0 Oct 28 16:02 ACF
-rwxrwxrwx 1 sbk6401 sbk6401 1465049 Oct 28 16:02 ACF.zlib
-rwxrwxrwx 1 sbk6401 sbk6401      30 Jun  4 09:14 flag.txt
$ cat flag.txt
flag{3L3ctr1c_5t33L_b1G_H41R}
```

Flag: `flag{3L3ctr1c_5t33L_b1G_H41R}`

Forensics

What's the Wallet

Description

> Ransomware was recently discovered on a system within De Monne's network courtesy of a DEADFACE member. Luckily, they were able to restore from backups. You have been tasked with finding the Bitcoin wallet address from the provided sample so that it can be reported to the authorities. Locate the wallet address in the code sample and submit the flag as flag{wallet_address}.

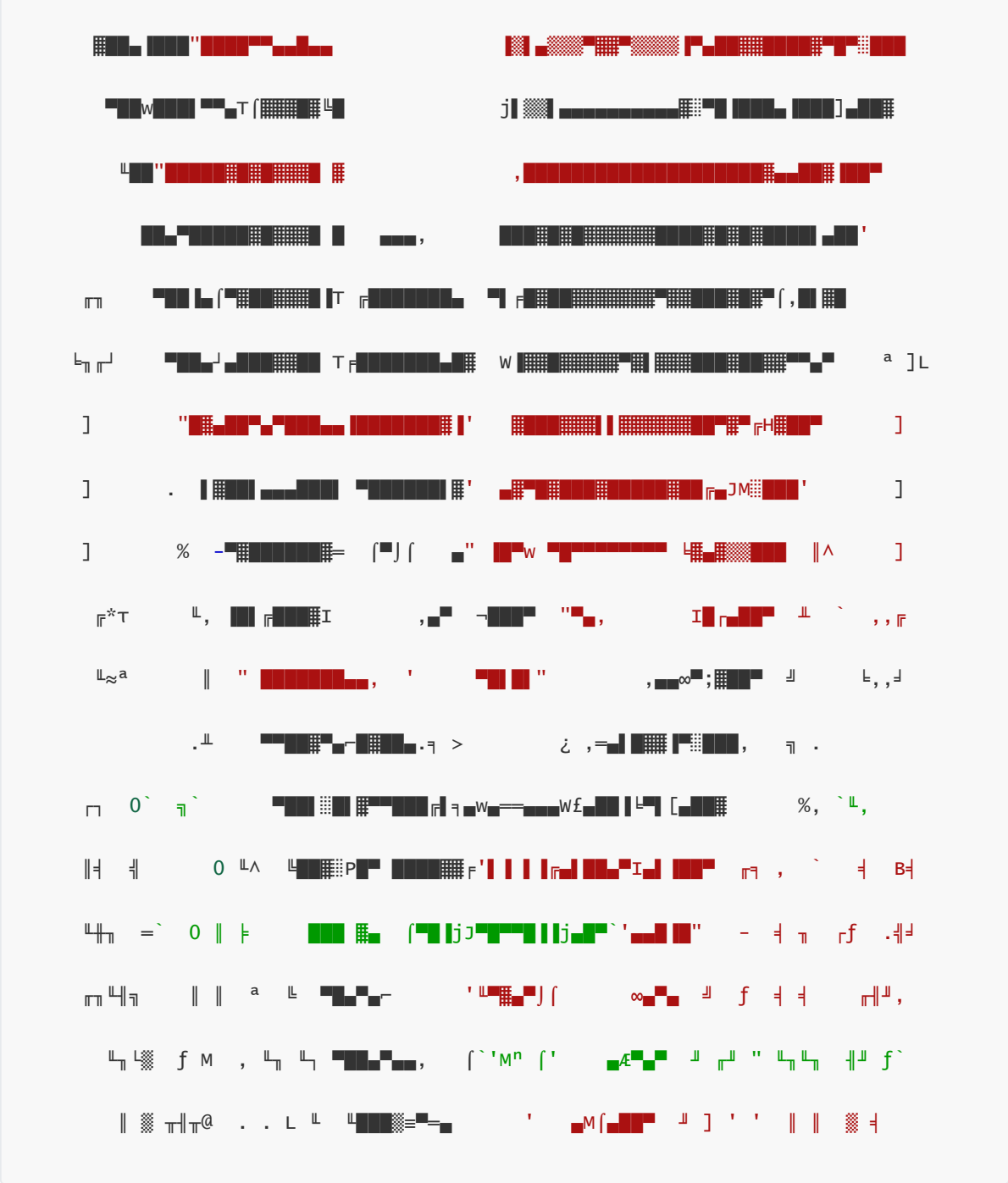
Source code

:::spoiler Source Code

```
# Get the current date and time
$currentTime = Get-Date

# Format the current date and time
$formattedTime = $currentTime.ToString("dddd, MMMM d, yyyy - HH:mm:ss")

# Get the current time in different time zones
$utcTime = Get-Date -Format "HH:mm:ss" -Utc
$nyTime = (Get-Date).ToUniversalTime().AddHours(-4).ToString("HH:mm:ss")
$tokyoTime = (Get-Date).ToUniversalTime().AddHours(9).ToString("HH:mm:ss")

# Display the current date and time
Write-Host "Current Date and Time: $formattedTime"
Write-Host "----------------------------------------"

# Display the current time in different time zones
Write-Host "Current Time (UTC): $utcTime"
Write-Host "Current Time (New York): $nyTime"
Write-Host "Current Time (Tokyo): $tokyoTime"

$encrypt = "djkulwiflwingsaili2ik7h5l2bn"
```

```
$encodedString =
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($encrypt))

Write-Host "Encoded String: $encodedString"

# Check if the system supports virtualization
$virtualizationEnabled = Get-WmiObject Win32_ComputerSystem | Select-Object -
ExpandProperty VirtualizationFirmwareEnabled

if ($virtualizationEnabled -eq $true) {
    Write-Host "Virtualization is enabled on this system."
} else {
    Write-Host "Virtualization is not enabled on this system."
}

# Check if Hyper-V is installed and enabled
$hyperVEnabled = Get-WindowsOptionalFeature -FeatureName Microsoft-Hyper-V-All -
Online | Where-Object { $_.State -eq "Enabled" }

if ($hyperVEnabled) {
    Write-Host "Hyper-V is installed and enabled on this system."
} else {
    Write-Host "Hyper-V is not installed or enabled on this system."
}

$encodedScript = @"
function Store-BtcWalletAddress {
    `$global:BtcWalletAddress =
[System.Convert]::FromBase64String([System.Text.Encoding]::UTF8.GetBytes('bjMzaGE
1bm96aXhlNnJyZzcxa2d3eWlubWt1c3gy'))
}

# Call the function to store the encoded Bitcoin wallet address
Store-BtcWalletAddress

# Access the stored encoded Bitcoin wallet address
Write-Host "Encoded Bitcoin Wallet Address: `$BtcWalletAddress"
"@

$encodedScriptBytes = [System.Text.Encoding]::Unicode.GetBytes($encodedScript)
$encodedScriptBase64 = [System.Convert]::ToBase64String($encodedScriptBytes)

Write-Host "Encoded Script:"
Write-Host $encodedScriptBase64

$address = "127.0.0.1"

# Set ping parameters
$pingCount = 4
$timeout = 1000  # Timeout value in milliseconds

# Perform the ping
$pingResult = Test-Connection -ComputerName $address -Count $pingCount -
TimeoutMilliseconds $timeout
```

```
if ($pingResult) {
    Write-Host "Ping to $address was successful."
    Write-Host "Ping statistics for $address :"
    Write-Host "-----------------------------------"
    Write-Host "Packets: Sent = $($pingResult.Count), Received =
$($pingResult.ReceivedCount), Lost = $($pingResult.Loss)"
    Write-Host "Approximate round trip times in milliseconds:"
    $pingResult | ForEach-Object {
        Write-Host "    Response from $($_.Address): time=$($_.ResponseTime)ms
TTL=$($_.TimeToLive)"
    }
} else {
    Write-Host "Ping to $address failed."
}

$apiUrl = "https://api.coingecko.com/api/v3/simple/price?
ids=bitcoin&vs_currencies=usd"
$response = Invoke-RestMethod -Uri $apiUrl
$bitcoinPrice = $response.bitcoin.usd
Write-Host "Current Bitcoin Price (USD): $bitcoinPrice"

$word = "Your data"
$encryptionKey = "MyEncryptionKey123"

$secureString = ConvertTo-SecureString -String $word -AsPlainText -Force
$encryptedData = ConvertFrom-SecureString -SecureString $secureString -Key (1..16
| ForEach-Object { $encryptionKey[$_] })
$encryptedString = [System.Text.Encoding]::UTF8.GetString($encryptedData)

Write-Host "Encrypted Word: $encryptedString"

$message = "YOur D@tA has been 3ncrypted pay 100 BTC (Bitcoin) to Deadface"
Write-Host $message
```



:::

Recon

這題雖然歸類為forensics但應該是reverseㄅ，有夠水，他只是要知道電子錢包的地址是啥，所以看一下source code就知道了

Exploit

```
>>> from base64 import *
>>> b64decode(b'bjMzaGE1bm96aXhlNnJyZzcxa2d3eWlubWt1c3gy')
b'n33ha5nozixe6rrg71kgwyinmkusx2'
```

Flag: `flag{n33ha5nozixe6rrg71kgwyinmkusx2}`

Host Busters 1

Description

> Turbo Tactical has gained access to a DEADFACE machine that belongs to gh0st404. This machine was used to scan one of TGRI's websites. See if you can find anything useful in the vim user's directory.
>
> On a side note, it's also a good idea to collect anything you think might be useful in the future for going after DEADFACE.
>
> Submit the flag as flag{flag_here}.
>
> vim@gh0st404.deadface.io
>
> Password: letmevim

Recon

這一題就比較有趣了，不過也是水題，就是個VimJail，可以看一下VimJail

```
$ ssh vim@gh0st404.deadface.io
vim@gh0st404.deadface.io's password:
:set shell=/bin/bash
:shell
vim@e5db30e81586:~$ ls -al
total 24
drwxrwxr-x 1 vim  vim  4096 Jul 29 23:05 .
drwxrwxr-x 1 root root 4096 Jul 29 23:05 ..
-rw-r--r-- 1 vim  vim   220 Apr 23 21:23 .bash_logout
-rw-r--r-- 1 vim  vim  3526 Apr 23 21:23 .bashrc
-rw-r--r-- 1 vim  vim   807 Apr 23 21:23 .profile
-rw-rw-r-- 1 vim  vim    26 Jul 29 23:05 hostbusters1.txt
vim@e5db30e81586:~$ cat hostbusters1.txt
flag{esc4P3_frOm_th3_V1M}
```

Flag: `flag{esc4P3_frOm_th3_V1M}`

Malum

Description

> Well, it happened. The ransomware event took us out but we are recovering. It's Tuesday now and time to head into the office. As you arrive your boss walks into the SOC with a sigh and look right to you; here we go. He drops a USB on your desk and says "I need you to go through all the logs to find out HOW these guys got valid credentials to attack us". Can you identify the threat vector that was used to gain persistence into the network by reading through security logs? What you find will be the flag.
>
> Submit the flag as flag{flagText}

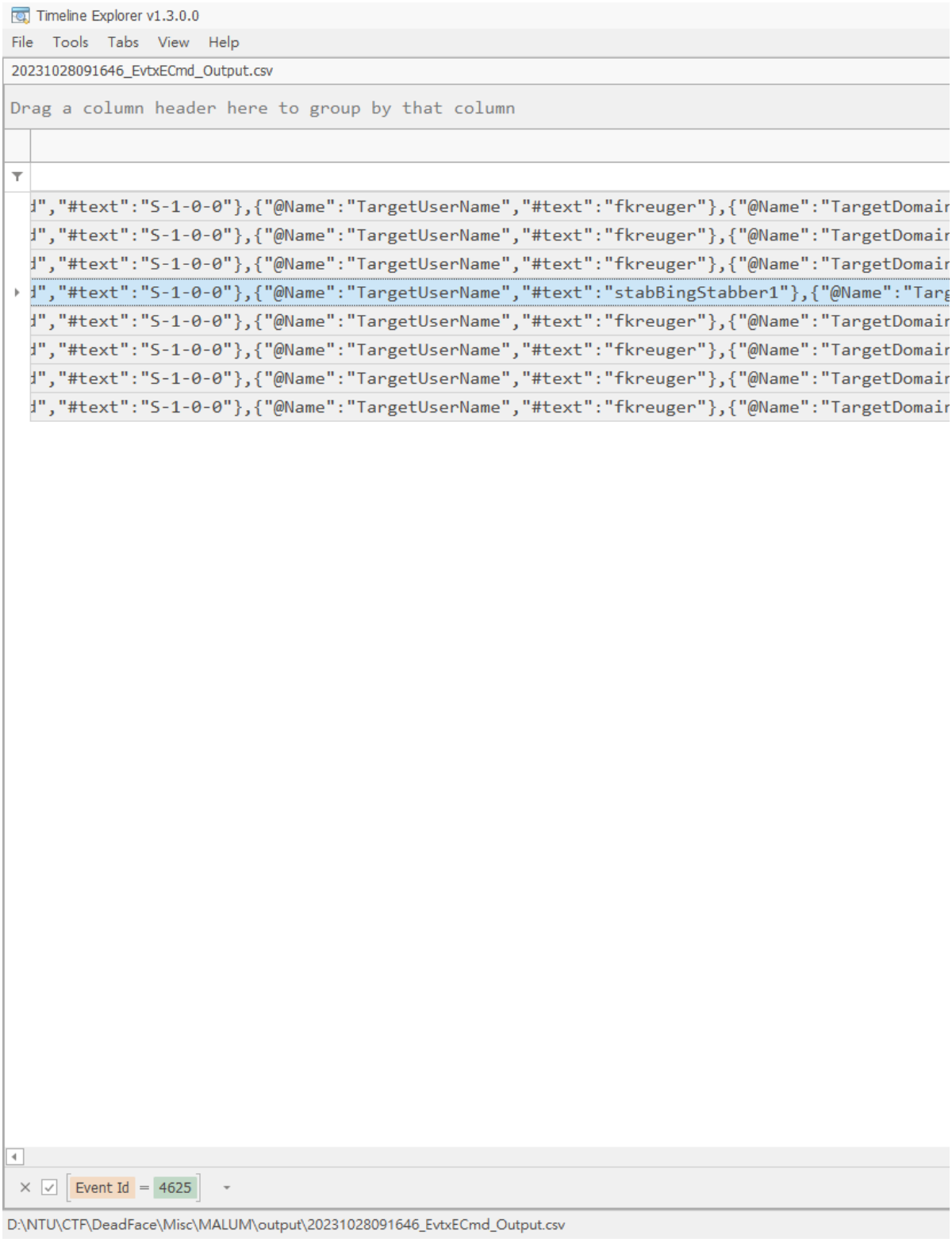
Recon

這一題有一點小通靈成分，剛好最近在玩windows forensics的東西所以直接用EvtxECmd.exe換成csv再用timeline explorer篩filter，如果單純用windows的event explorer，眼睛會脫窗

Exploit

```
$ EvtxECmd.exe -f Maybehere.evtx --csv output
```

首當其衝一定先看event ID 4625(logon failure)，就變成非常少的event，看一下裡面的payload，就會發現奇怪的字串，想說試看看結果就猜中了，解的莫名其妙QAQ



Flag: `flag{stabBingStabber1}`

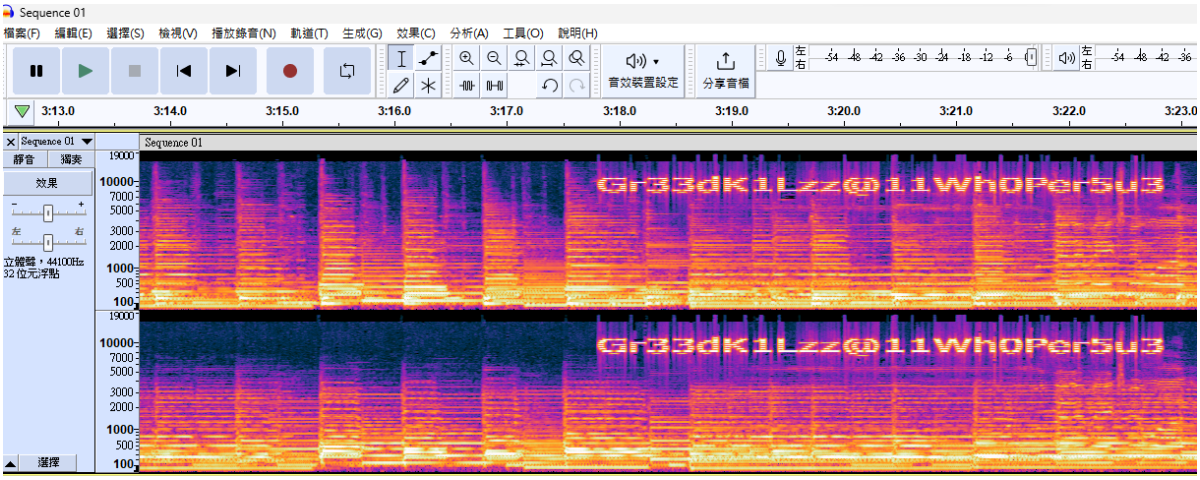
Tin Balloon

Description

> We've discovered that DEADFACE was somehow able to extract a fair amount of data from Techno Global Research Industries. We are still working out the details, but we believe they crafted custom malware to gain access to one of TGRI's systems. We intercepted a Word document that we believe mentions the name of the malware, in addition to an audio file that was part of the same conversation. We're not sure what the link is between the two files, but I'm sure you can figure it out!
>
> Submit the flag as: flag{executable_name}. Example: flag{malware.exe}.

Recon

這一題真的是氣死我了，被0和O搞到咪咪冒冒
看到音檔直覺就是audacity開頻譜，果然看到字串藏在裡面，感覺就是word file的password，到這邊大概也只過五分鐘，心想可以秒解，殊不知被搞心態



Word Password: `Gr33dK1Lzz@11WhOPer5u3`
Word Content

> We have the ID card of one the brand new employees Alejandro, We now know the location of Techno Global, we have a man on sight that has been tailing him. We believe we can get into the facility at 3 am.
> We don't know how long we can have a foothold on the system but we are going to use Wh1t3_N01Z3.exe to sent out a reverse shell. Be prepared to listen for the signal.

Flag: `flag{Wh1t3_N01Z3.exe}`

Host Busters 2

Description

> Now that you've escaped out of vim, scope out and characterize the machine. See if there are any other flags you can find without having to escalate to another user.
>
> Submit the flag as flag{flag_here}.
>
> vim@gh0st404.deadface.io
>
> Password: letmevim

Recon

這一題是參考[Host Busters 2 - WP](#)這一篇，主要的思路是:

1. 查看目前正在執行的process
2. 發現有一個其實是有關flag的udp process
3. 用netstat去聽拿到flag

Exploit

1. `$ ps aux`

```
vim@1329c5769906:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
vim            1  0.0  0.0   2576   928 pts/0    Ss   08:44   0:00 /bin/sh
/usr/bin/start
vim            9  0.0  0.0   1036   744 pts/0    S    08:44   0:00
/usr/bin/srv
vim           10  0.3  0.2  11692  9120 pts/0    Sl   08:44   0:12 /bin/vim
/home/gh0st404/config
vim           12  0.0  0.0   4188  3496 pts/0    S    08:45   0:00 /bin/bash
root          21  0.0  0.0  15404  1324 ?        Ss   08:45   0:00 sshd:
/usr/sbin/sshd [listener] 0 of 10-100 star
vim           38  0.0  0.0   8088  3912 pts/0    R+   09:35   0:00 ps aux
```

從結果可以看到`/usr/bin/start`這個script被執行，然後`/usr/bin/srv`這個script直覺應該是和運行該題目的server有關係，從文章中作者有提到`/usr/bin/srv`其實是被UPX packer包起來的東西，所以沒有任何有關flag的plaintext可以從裡面撈

```
vim@1329c5769906:~$ cat /usr/bin/start
#!/bin/sh
/usr/bin/srv &
#/etc/init.d/ssh start
/bin/vim /home/gh0st404/config
exit 0
```

2. 不過從`/usr/bin/start`中的內容來看，應該是執行`/usr/bin/srv`就開啟這個題目了，所以我們可以直接看他正在跑的網路狀況(netstat)

```
vim@1329c5769906:~$ netstat -plano
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
  PID/Program name     Timer
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
   -                    off (0.00/0/0)
tcp6       0      0 :::22                   :::*                    LISTEN
   -                    off (0.00/0/0)
udp        0      0 0.0.0.0:9023            0.0.0.0:*
  9/srv                off (0.00/0/0)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name
  Path
```

從結果可以知道有一個localhost使用udp protocol的連線正在執行，所以我們可以直接用nc戳9023 port，直接用-u(udp mode)傳送資料過去

```
vim@1329c5769906:~$ nc -u 0.0.0.0 9023

flag{Hunt_4_UDP_s3rv3r}
```

Flag: `flag{Hunt_4_UDP_s3rv3r}`

Traffic Analysis

Sometimes IT Lets You Down

Description

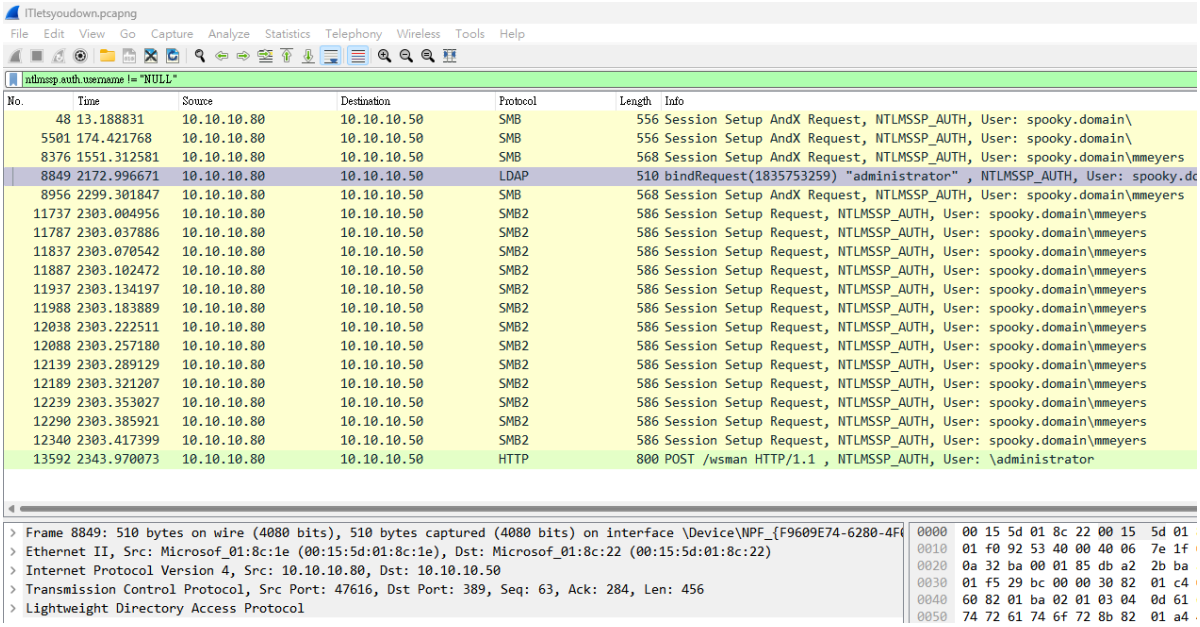
> Lytton Labs has been having a slew of weird activity in the network lately. This recent PCAP capture we know contains a user account who compromised our domain controller. Can you figure out what user account was compromised?
>
> Submit the flag as: flag{username}.

Recon

因為最近在玩一些cyberdefender的traffic analysis，其實這一題也和之前寫的[CyberDefender - PsExec Hunt](蠻像的，我也沒有仔細研究這個traffic大致在幹嘛，所幸直接看ntlm的authenticated username去撈就知道了

Exploit

Filter: `ntulssp.auth.username != "NULL"`



Flag: `flag{mmeyers}`

UVB-76 (Hello, are you there?)

Description

> Lytton Labs system administators are talking through network traffic, probably complaining about the Turbo Tactical assessment. I have looked and looked but I can't figure it out. Can you find the secret message?

Exploit

水題，直接strings search "flag"就找到了

Flag: `flag{is_this_thing_on?}`

Programming

Dead Drop

Description

> The Incident Response Team at Aurora Pharmaceuticals recently recovered this file from a user's computer. The artifacts indicate it was accessed by what they believe to be multiple DEADFACE members. The program appears to have set up the user's workstation as a dead drop for DEADFACE members to convert a secret numerical code into a password string for further target access. Our decoding attempts have been unsuccessful, but the script appears to contain a recovery code that may be a good starting point.
>
> Submit the flag as flag{the password} exactly how print_password() returns it.

Source Code

```
# Password recovery:
# buA9kvZ=T_A}b[J8l:@ob_tviPZtb_<olOpxkvZ=T_=xju]olOpxkvZ=T_bxlu]olOpxkvZ=QIEE
arr = ['empty', 'interest', 'current', 'valuable', 'influence', 'from',
'scolded', 'would', 'got', 'key', 'facility', 'run', 'great', 'tack', 'scent',
'close', 'are', 'a', 'plan', 'counter', 'earth', 'self', 'we', 'sick', 'return',
'admit', 'bear', 'cache', 'to', 'grab', 'domination', 'feedback', 'especially',
'motivate', 'tool', 'world', 'phase', 'semblance', 'tone', 'is', 'will', 'the',
'can', 'global', 'tell', 'box', 'alarm', 'life', 'necessary']
def print_password(nums):
    if len(nums) < 1:
        print("Must provide a list of at least one number i.e. [1]")
    return ("flag{{{}}}".format(" ".join([arr[num] for num in nums])))


def left_shift(s, n):
    return ''.join(chr(ord(char) - n) for char in s)
```

Recon

看了第一個hint才有點頭緒，先用他提供的left_shift function，先看有沒有特別的部分，忽然看到貌似base64，忽然就拿到flag了

Exploit

```
ct =
'buA9kvZ=T_A}b[J8l:@ob_tviPZtb_<olOpxkvZ=T_=xju]olOpxkvZ=T_bxlu]olOpxkvZ=QIEE'

from base64 import b64decode
for i in range(50):
    guess = left_shift(ct, i)
    try:
        guess = b64decode(guess).decode('utf-8')
        print(guess)
    except:
        pass

pt = [41, 2, 18, 39, 35, 30]
print(print_password(pt))
```

Flag: `flag{the current plan is world domination}`

Chatty Cathy

Description

> That Python dead drop program put Aurora Pharmaceuticals' IT team on high alert; they looked closer at notifications in their Security Operation Center (SOC) and identified even more malware deep in the network! They retrieved a compiled binary along with some rough code from yet another infected user workstation. It appears to be some sort of Command and Control server, but requires a password to be accessed. Intel indicates that the full flag from Dead Drop may work as the password.. Dig into this program and see what secrets it holds.

> Submit the flag as: flag{flag text here}.

Source Code

:::spoiler Source Code

```
typedef struct {
    struct in_addr address;
    unsigned short port;
    unsigned short start_cmd;
} settings;

void check_pass(){
    char password[60];
    printf("Enter password: ");
    scanf("%59[^\n]s", password);
    // TODO: Check password
}

settings* get_settings(){
    settings *serv_settings = (settings*)malloc(sizeof(settings));
    serv_settings->start_cmd = 0;

    printf("Enter Command Number: ");
    scanf("%hu", &(serv_settings->start_cmd));
    if(serv_settings->start_cmd != 0){
        printf("Invalid command number.\n");
        exit(0);
    }

    printf("Enter Listening Address: ");
    char addr[16];
    scanf("%15s", addr);
    inet_pton(AF_INET, addr, &(serv_settings->address));

    printf("Enter Listening Port: ");
    scanf("%lu", &(serv_settings->port));
    return serv_settings;
}

int setup_socket(settings *serv_settings){
    int listenfd = 0;
    listenfd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in serv_addr;
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(serv_settings->port);
    serv_addr.sin_addr = serv_settings->address;

    int fd = socket(AF_INET, SOCK_STREAM, 0);
    bind(fd, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
    return fd;
}
```

```c
void print_flag(){
    // TODO: Print out flag.
}

void record_message(char *message){
    FILE *fp = fopen("./log.txt", "a");
    fprintf(fp, message);
    fclose(fp);
}

void print_error(){
    printf("The server received a message from the client but is unable to
display it.\n");
}

void handle_message(unsigned short start_cmd, char *message){
    switch(start_cmd){
        case 0:
            record_message(message);
            break;
        case 1:
            print_flag();
            break;
        default:
            print_error();
    }
}

void accept_conns(int socket, settings *serv_settings){
    bind(socket, (struct sockaddr *)&serv_settings, sizeof(serv_settings));
    listen(socket,5);

    char address[INET_ADDRSTRLEN];
    inet_ntop(AF_INET, &(serv_settings->address), address, INET_ADDRSTRLEN);
    printf("Listening for connections on %s:%hu\n", address, serv_settings-
>port);

    int conn;
    char message[100] = "";
    while(conn = accept(socket, (struct sockaddr *)NULL, NULL)) {
        int pid;
        if((pid = fork()) == 0) {
            while (recv(conn, message, 100, 0)>0) {
                handle_message(serv_settings->start_cmd, &message);
                *message = "";
            }
            exit(0);
        }
    }
}

int main() {
    check_pass();
    settings *serv_settings = get_settings();
    int socket = setup_socket(serv_settings);
    accept_conns(socket, serv_settings);
```

```
    free(serv_settings);
    printf("Complete\n");
    return 0;
}
```

:::

Recon

這一題有點微妙，觀念是簡單的，連pwn都算不上，但還是pwn的觀念，基本上要從這一支程式拿到flag，在有source code的情況下甚麼都好說，不然以這種狀態要找到print flag的function真的要逆到死

Exploit

我的作法是直接不管他所有的check然後跳到print_flag function讓他直接吐flag給我這樣

```
$ gdb not_c2_server
gef➤  r
Starting program: /mnt/d/NTU/CTF/DeadFace/Misc/Chatty Cathy/not_c2_server
Enter password: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Breakpoint 1, 0x000000000040176c in ?? ()
gef➤  j *0x40194A
Continuing at 0x40194a.
flag{heaps and stacks and bugs oh my!}

Program received signal SIGSEGV, Segmentation fault.
0x00000000004019d7 in ?? ()
```

Flag: `flag{heaps and stacks and bugs oh my!}`

The CDR of the CAR... RAH, RAH, RAH!!!

Description

:::spoiler

> The LISP programming language (which stands for “Lots of Insane and Stupid Parentheses”) was used as an early form of list processing. There was even a “LISP COMPUTER” where the assembly language was LISP!
>
> LISP was famous (infamous?) for its numerous parentheses. Miss one, and the whole program fails!
>
> LISP had among its data types “atoms” (single items) and “lists” (multiple items), formatted like this:
>
> ```
> (apple tomato (grape bear (banana)) ((President Trump),(President Obama)))
> ```
>
> Two of LISP’s most famous functions were car (which returns the first item in a list), and cdr (which returns all but the first item in a list). They could be used, together with recursion, to perform loops. In fact, early LISP had no native looping structure other than recursion.

> Attached to this challenge is a flat list of words. To obtain the flag, create a program in Python that calls emulated car and cdr functions (already provided for you) to produce the correct list. The list has to be grouped into a list of atoms and lists such that the program, as described by the Lytton, IN High School Basketball Cheerleaders, produces the correct list. Use their cheer to lay out the function calls in a series of nested calls...
>
> The cheer goes like this:
>
> The CDR (1) of the CAR!
> The CDR of the CAR!
> The CAR of the CDR of the CDR of the CAR!
> The CAR of the CDR of the CDR of the CAR (12)!
>
> Hence, the first CDR (1) is the outermost call, and the CAR (12) is the innermost call. The single parameter is the complete list of words in the wordlist, grouped appropriately to produce the output list.
>
> ```
> (1)                 (12)
> cdr(car(cdr(car...  car(('fish', ('vermin','blatant',('ascent'...))))
> ```
>
> Here is the input wordlist as a flat list, without grouping, as well as the expected output and instructions for submitting your answer to obtain the flag.
>
> Submit your answer to the server like this:
>
> echo "car(cdr(("blah", "blah")))" | nc -nv 143.198.226.223 50000
> :::

Recon

這一題很特別，應該算是目前比較難的題目，他主要是介紹了LISP這個語言，然後用他指定的兩個function(CAR/CDR)輸出特定的list，CAR主要return了第0個item，而CDR是return除了第0個以外的其他item，用python寫就會是以下這樣:

```
def car(a): return a[0]
def cdr(a): return a[1:]
```

然後他已經定義了一套function的call stack應該長怎樣:
`cdr(car(cdr(car(car(cdr(cdr(car(car(cdr(cdr(car(List)))))))))))`
而預期的輸出應該是: `('pugnacious', 'wallaby', 'savant', 'zarf')`
所以重點是要如何設計中間的List，另外他有給initial phrase

```
('ascent','xray','yarbrough','jackal','minstrel','nevermore','outcast','kitten','
victor','pugnacious','wallaby','savant','zarf','tango','ultimatum','papyrus','qui
ll','renegade','llama','ghost','hellscape','industrious','zombification','bestial
','cadre','dark','efficacious','foundational')
```

順序不能調，但中間要如何增減括號都無所謂，所以我設計了一套自己的演算法:
從最開頭往回看，也就是`cdr`→`car`→`cdr`→`car`...

- 只要看到`cdr`，則只需要括前面的item
- 若碰到`car`，則圈後面，並包含前一個狀態所有東西包起來

- 如果碰到錯誤，應該就是讓串入function的參數變成兩個以上，此時只要把payload全部加上括號就可以了

這樣才可以按照順序慢慢拆回來變成我們要的樣子，演算法這樣設計應該算是top-down的模式，至於圈多少就各自判斷，只要判斷完後可以剛好圈完就好，因為`cdr`主要是往前圈，`car`是往後圈，所以我會判斷，`cdr`和`car`各有幾個，然後分配一下每一次要圈多少

Exploit - Try & Error

1. Initial State
   因為我們知道最後要的是中間的`('pugnacious', 'wallaby', 'savant', 'zarf')`，所以第一個遇到的是`cdr`，他會取第一個以外其他的items，所以我們要圈前面的item，會變成這樣:
   `('victor','pugnacious', 'wallaby', 'savant', 'zarf')`
2. 遇到`car`要圈後面，再包含前一個狀態全部包起來，也就是會變成
   `(('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus'))`
3. 遇到`cdr`則只需要圈前面的item，會變成`(('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus'))`
4. 遇到`car`就會是`(('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')`
5. `((('outcast','kitten'),('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
   `('ghost','hellscape','industrious')`
6. `('minstrel','nevermore'),((('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
   `('ghost','hellscape','industrious')`
7. `('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
   `('ghost','hellscape','industrious')`
8. `(('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
   `('ghost','hellscape','industrious')),('zombification','bestial','cadre')`
9. `((('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),`
   `('victor','pugnacious','wallaby','savant','zarf'),`
   `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
   `('ghost','hellscape','industrious')),('zombification','bestial','cadre')),`
   `('dark','efficacious')`
10. `('xray'),((('yarbrough','jackal'),('minstrel','nevermore'),`
    `((('outcast','kitten'),('victor','pugnacious','wallaby','savant','zarf'),`
    `('tango','ultimatum','papyrus')),('quill','renegade','llama')),`
    `('ghost','hellscape','industrious')),('zombification','bestial','cadre')),`
    `('dark','efficacious')`

11. `('ascent'),('xray'),((('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),('victor','pugnacious','wallaby','savant','zarf'),('tango','ultimatum','papyrus')),('quill','renegade','llama')),('ghost','hellscape','industrious')),('zombification','bestial','cadre')),('dark','efficacious')`
12. `((('ascent'),('xray'),((('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),('victor','pugnacious','wallaby','savant','zarf'),('tango','ultimatum','papyrus')),('quill','renegade','llama')),('ghost','hellscape','industrious')),('zombification','bestial','cadre')),('dark','efficacious')),('foundational'))`

```
$ echo "cdr(car(cdr(car(car(cdr(cdr(car(car(cdr(cdr(car((((('ascent'),('xray'),
((('yarbrough','jackal'),('minstrel','nevermore'),((('outcast','kitten'),
('victor','pugnacious','wallaby','savant','zarf'),
('tango','ultimatum','papyrus')),('quill','renegade','llama'),
('ghost','hellscape','industrious'))),('zombification','bestial','cadre')),
('dark','efficacious')),('foundational')))))))))))))))"  | nc -nv 143.198.226.223
50000
Connection to 143.198.226.223 50000 port [tcp/*] succeeded!
Send me your answer...
You are CORRECT!!! flag{BABY_you_can_DRIVE_my_CDR!!!}
```

其實答案有很多種，除了圈多圈少以外，圈法也有很多種，只要local端可以過，愛怎麼圈就怎麼圈

Flag: `flag{BABY_you_can_DRIVE_my_CDR!!!}`

Crypto

Letter Soup

Description

> We believe we have ran into one of the newest members of DEADFACE while they were waiting for the train. The member seemed to have gotten spooked and stood up suddenly to jump on the train right before the doors shut. They seemed to have gotten away, but dropped this innocent looking word search. I believe this member might be actually a courier for DEADFACE. Let's solve the word search to decode the mystery message. We believe the message might tell us their next move.
>
> Submit the flag as flag{TARGETNAME} (e.g., flag{THISISTHEANSWER})

Recon

這一題想了很久，明明是低分的題目卻沒有其他想法(應該說很多想法卻屢屢碰壁)，最後是看其他人的WP才恍然大悟，其實和一開始想得差不多，就是把填字遊戲完成後把沒有圈到的字由左到右集合起來再用ROT的方式找flag，就是最後的步驟卡住，其實有點misc的感覺，蠻新鮮的?

Flag: `flag{ASBLACKFEATHERSSHINEINTHESUN}`

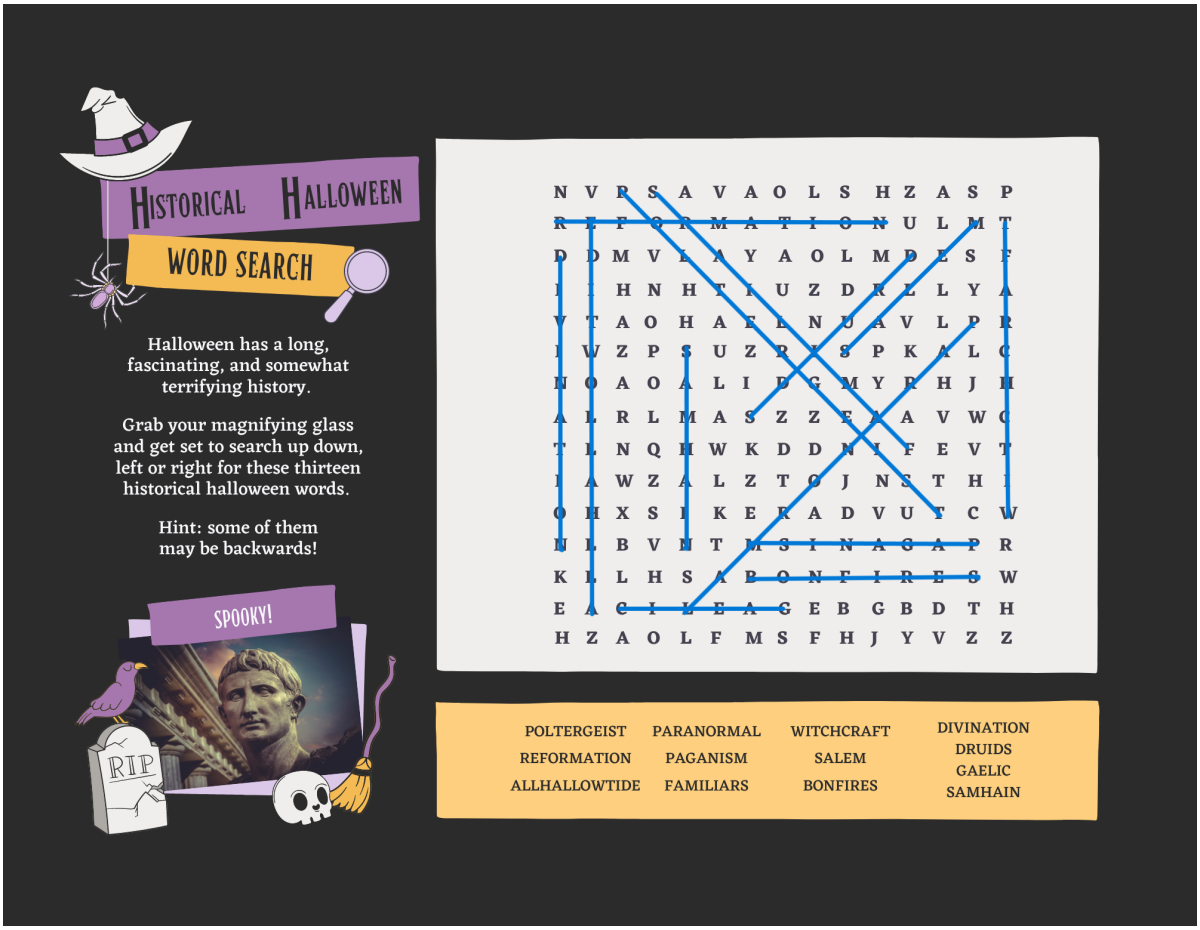
Refill on Soup

Description

> How could we have missed this?? There were TWO word searches stuck together that the DEADFACE courier dropped. We've already solved the first one, but maybe solving this second word search will help us uncover the secret message they're trying to covertly relay to the other members of DEADFACE. Hopefully, THIS will tell us how they plan to execute their next move.
>
> Submit the flag as flag{TARGETNAME} (e.g., flag{THISISTHEANSWER})

Recon

和上一題一模一樣的操作，解密之後會是: `GO TO THE LAST LINE FOR THE FLAG ANSWER THAT GOES INSIDE THE BRACKETS STOP GJPDWWXOPSESMCGMAQLDXTWONVUOMKDEALPXXUZUWMA ASTHEYFLYACROSS`



Flag: `flag{ASTHEYFLYACROSS}`

Exploit

Reference