CAPTURE THE FLAG

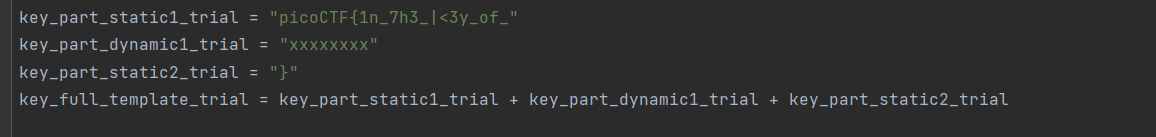
TASK: 3

1. REVERSE ENGINEERING:

1.Keygenme-py

*#CHALLENGE*

<https://mercury.picoctf.net/static/9055e7d35f5f4646338a1734aea0dda5/keygenme-trial.py>

*#SOLUTION*

1. From the given python code given in the question we can conclude that the flag contains three parts of which dynamic part is encrypted.

A computer screen shot of a program code

Description automatically generated

2. The check\_key functions contain the code that fills in the key\_part\_dynamic1\_trial. It takes the hexdigest of the sha256 hash of b"FRASER" and then selects a certain character by an indexing to a certain point on that string.

3. Thus we take a variable temp where we store the hash digest of the string b”FRASER”.

4. Now as we want the characters from index no. 4,5,3,6,2,7,1,8 and join them.

5. Thus we obtain our flag by joining the static and dynamic part of the code.

*A screenshot of a computer

Description automatically generated*

*#FLAG*

picoCTF{1n\_7h3\_|<3y\_of\_ac73dc29}

2.GDB BABY STEP 1

*#CHALLENGE*

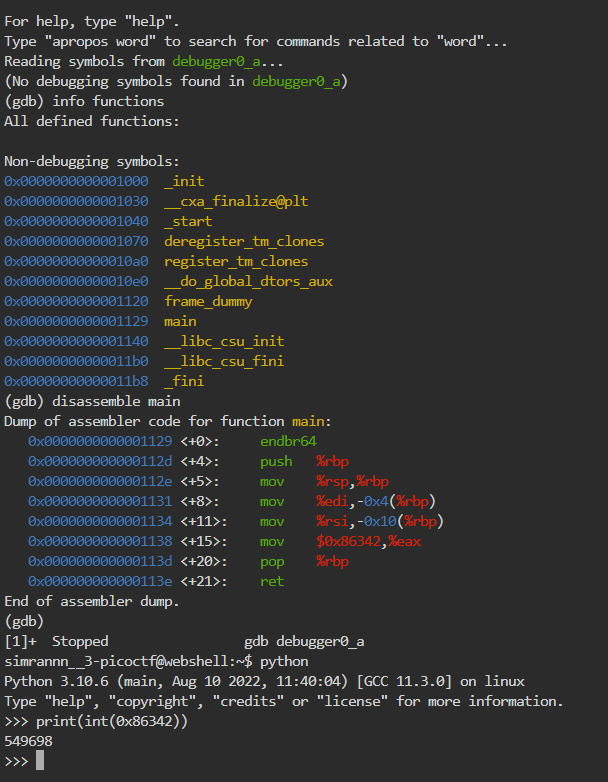
Can you figure out what is in the eax register at the end of the main function? Put your answer in the picoCTF flag format: picoCTF{n} where n is the contents of the eax register in the decimal number base. If the answer was 0x11 your flag would be picoCTF{17}. Disassemble [this](https://artifacts.picoctf.net/c/512/debugger0_a).

*#SOLUTION*

1. First, we list the contents of the file using **ls** command.
2. Next, we check the file type of debugger0\_a. We find its a ELF file, 64 bit Executable and not stripped.
3. Thus, we use **gdb** command.
4. Now next we need to locate the main function so for that we use **info functions** command. After locating we need to disassemble the assembly code of the main function in this file, we use the following command. “**disassemble main”.**
5. We need to search for the contents of the "**eax**" register. The contents are in hexadecimal ("**0x86342**").
6. Using python, we can easily convert the hexadecimal number to decimals. print(int(0x86342)) We get the number **549698** as decimal. Hence, the pico flag is **picoCTF{549698}.**

A black background with a black border

Description automatically generated with medium confidence



*#FLAG*

**picoCTF{549698}**

3.ARMssembly 0

*#CHALLENGE*

What integer does this program print with arguments 266134863 and 1592237099? File: [chall.S](https://mercury.picoctf.net/static/104d6022bcea93f53083aeb61b134e8b/chall.S) Flag format: picoCTF{XXXXXXXX} -> (hex, lowercase, no 0x, and 32 bits. ex. 5614267 would be picoCTF{0055aabb}).

*#SOLUTION*

1. We need to just convert the bigger number into the hexadecimal format se we do that using python .



*#FLAG*

picoCTF{6d1d2dd1}

2.BINARY EXPLOITATION:

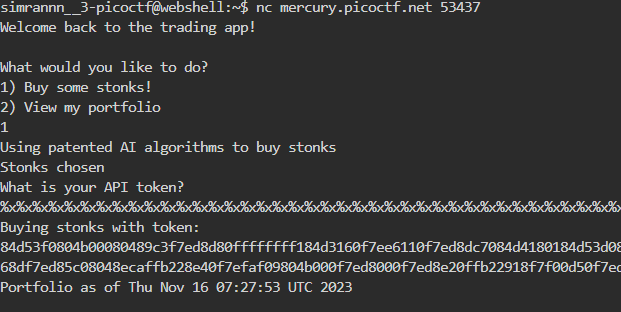
1.Stonks

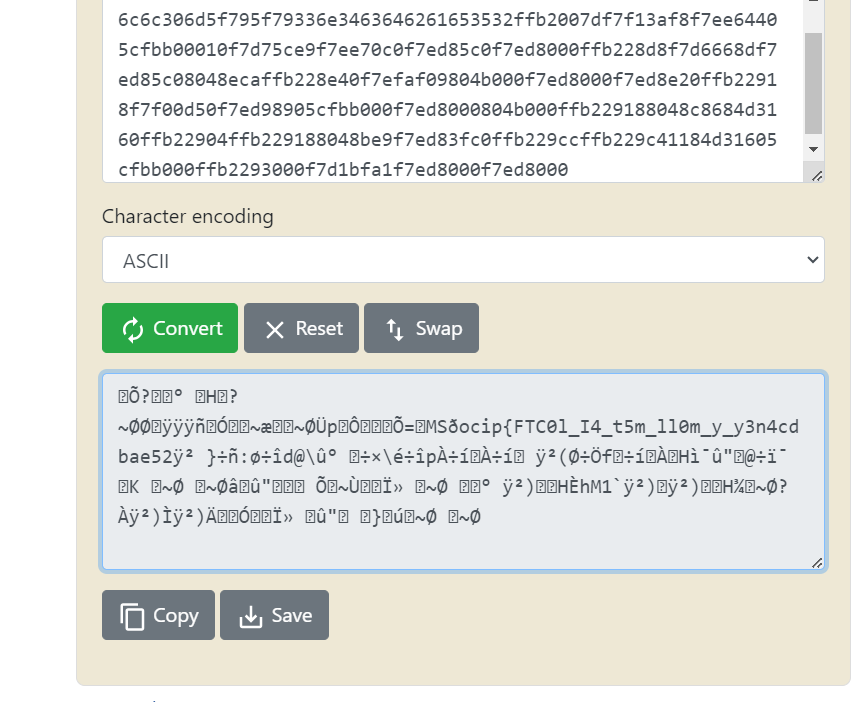
*#CHALLENGE*

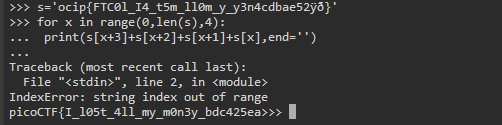
<https://mercury.picoctf.net/static/7e71fc0d8cc3339bfad6bf408f7dc510/vuln.c>

*#SOLUTION*

1. First, we download the vuln.c file and connected to the given server. Look at the **user\_buf** field. Upon running the program from the terminal, we found some hexadecimal values.
2. Next using online converter, we convert the hexadecimal values to ascii text.
3. To get string in correct order we use python.

**

**



*#FLAG*

picoCTF{I\_l05t\_4ll\_my\_m0n3y\_bdc425ea}

ÿ2

2.babygame01

*#CHALLENGE*

Get the flag and reach the exit.

Additional details will be available after launching your challenge instance.

*#SOLUTION*

3.WEB EXPOLITATION

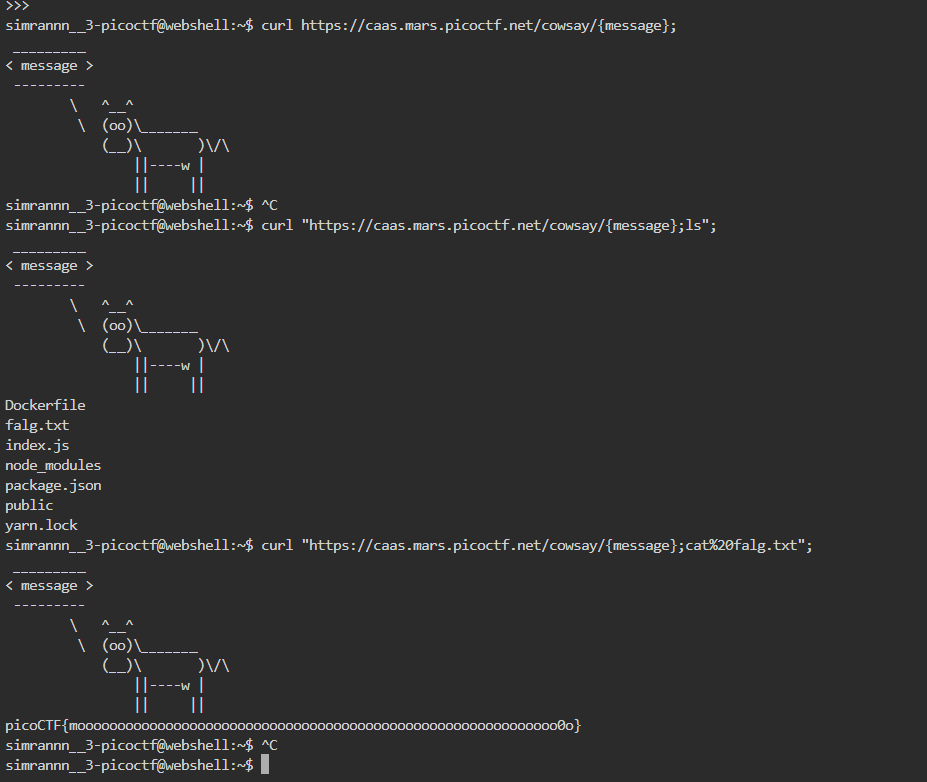
1.Caas

*#CHALLENGE*

<https://artifacts.picoctf.net/picoMini+by+redpwn/Web+Exploitation/caas/index.js>

*#SOLUTION*

1. First visit the provided URL for the CAAS challenge and explored it.Use the curl command in the terminal to visit the provided URL.
2. Now next we use the **ls** command to list the files.
3. Next, we can see there is a flag file which contains our flag thus we use the **cat** command.



*#FLAG*

picoCTF{moooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo0o}

2.Forbidden Paths

*#CHALLENGE*

Can you get the flag?Here's the website. We know that the website files live in /usr/share/nginx/html/ and the flag is at /flag.txt but the website is filtering absolute file paths. Can you get past the filter to read the flag?

*#SOLUTION*

1. First, we visit the provided URL for the Forbidden Paths challenge and explore the functionality of the web application.
2. Next, we know that the files are present in the **"/usr/share/nginx/html/"** directory, and the flag.txt file is present in the root directory. So, to reach the root directory, and display the flag.txt file, we can use the following input: **../../../../flag.txt**
3. After using this we get our required flag.

A screenshot of a computer

Description automatically generatedA blue background with black text

Description automatically generated

*#FLAG*

picoCTF{7h3\_p47h\_70\_5ucc355\_e5a6fcbc}

3.Local Authority

*#CHALLENGE*

Can you get the flag? Go to this [website](http://saturn.picoctf.net:50920/) and see what you can discover.

*#SOLUTION*

1. First we go to the login link which is <http://saturn.picoctf.net:50920/login.php>
2. Then on right clicking we get an option of **inspect.** On going on inspect option we have some files under it.
3. On opening secure.js file we get the code where we can find the required credentials that is username=’**admin**’ and password=’ **strongPassword098765**’.
4. Now on logging in we get our required flag.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A blue background with black text

Description automatically generated

*#FLAG*

picoCTF{j5\_15\_7r4n5p4r3n7\_05df90c8}

4.FORENSICS

1. tunn3l v1s10n

*#CHALLENGE*

We found this [file](https://mercury.picoctf.net/static/7b2d7c26630e977197022d0af09e3aeb/tunn3l_v1s10n). Recover the flag.

*#SOLUTION*

A screenshot of a computer screen

Description automatically generated