

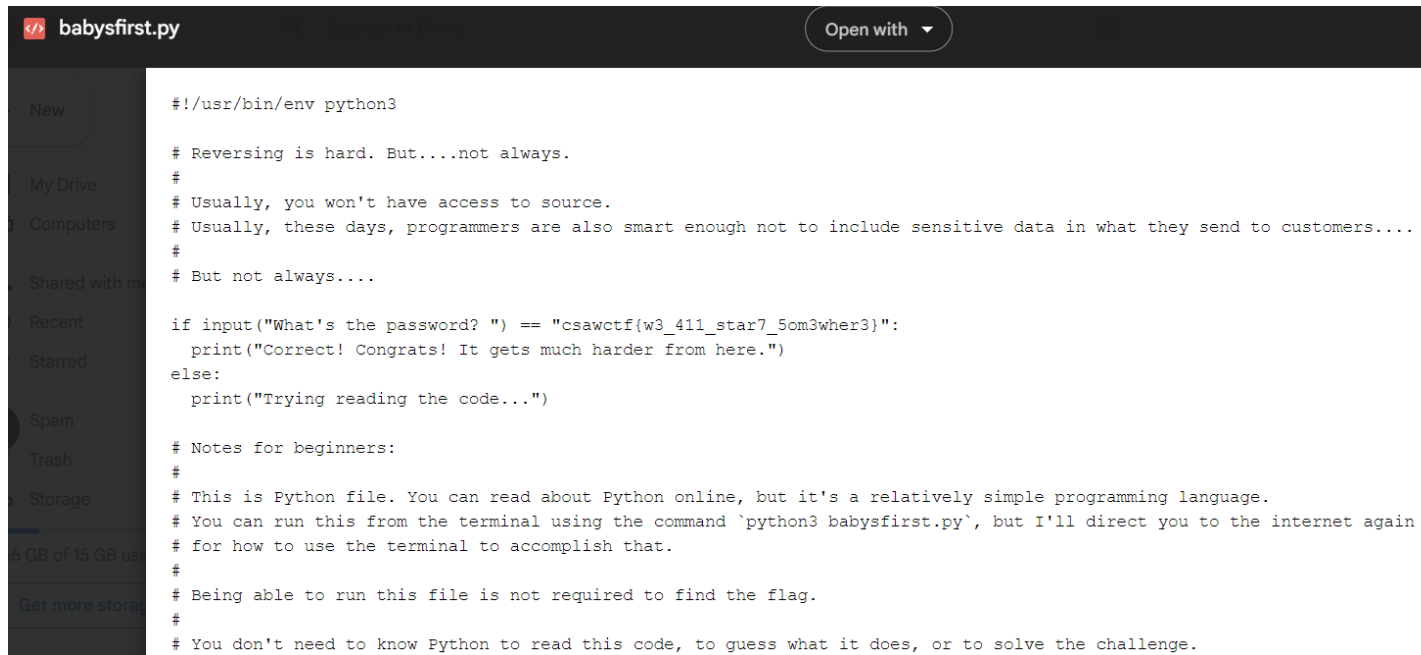
Team A&N
Niki Pola - np2356
Ashley Simons - aes841
LLM Challenge

Baby's first

10 Points

In this warm up CTF we are given two files: "babysfirst.py" and "challenge.json"

"babysfirst.py" In this Python file we are given an if statement that looks like it accepts the flag as the password. We can see the flag/password is hard coded into the code. But is this the flag? Let us confirm.



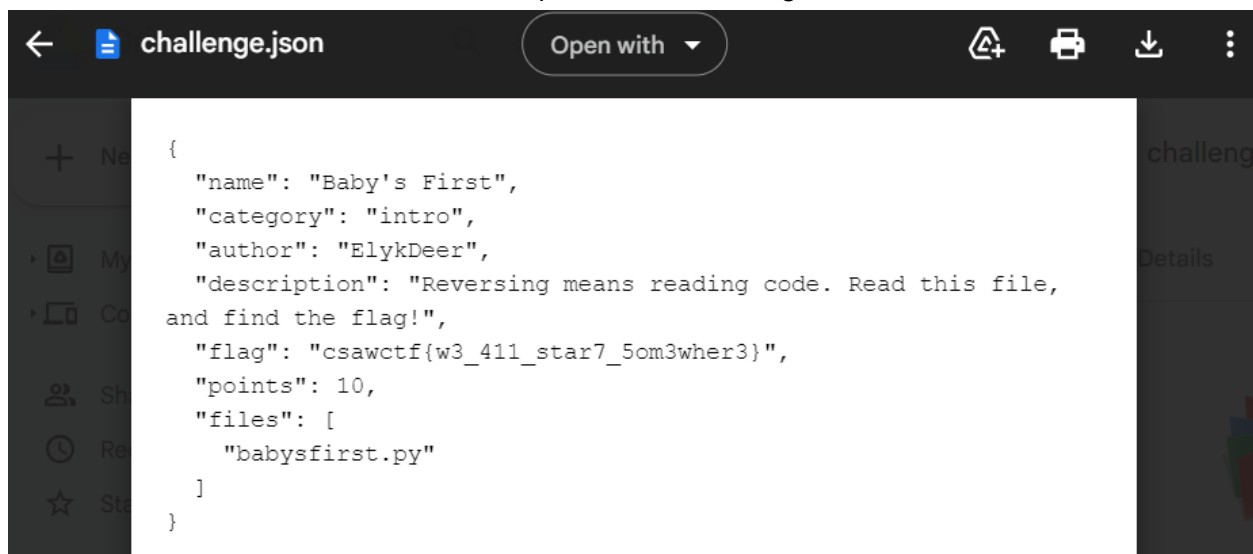
```
#!/usr/bin/env python3

# Reversing is hard. But....not always.
#
# Usually, you won't have access to source.
# Usually, these days, programmers are also smart enough not to include sensitive data in what they send to customers....
#
# But not always....

if input("What's the password? ") == "csawctf{w3_411_star7_5om3wher3}":
    print("Correct! Congrats! It gets much harder from here.")
else:
    print("Trying reading the code...")

# Notes for beginners:
#
# This is Python file. You can read about Python online, but it's a relatively simple programming language.
# You can run this from the terminal using the command `python3 babysfirst.py`, but I'll direct you to the internet again
# for how to use the terminal to accomplish that.
#
# Being able to run this file is not required to find the flag.
#
# You don't need to know Python to read this code, to guess what it does, or to solve the challenge.
```

"challenge.json" In this json file we are given generic information about the challenge. There is a "flag" section which looks like the password we saw in the babysfirst.py. By comparing the two files, we can confirm that the hard coded password is the flag.



```
{
  "name": "Baby's First",
  "category": "intro",
  "author": "ElykDeer",
  "description": "Reversing means reading code. Read this file, and find the flag!",
  "flag": "csawctf{w3_411_star7_5om3wher3}",
  "points": 10,
  "files": [
    "babysfirst.py"
  ]
}
```

FLAG: csawctf{w3_411_star7_5om3wher3}



This is a CTF challenge? What is the Flag?



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

```
# Reversing is hard. But....not always.
```

```
#
```

```
# Usually, you won't have access to source.
```

```
# Usually, these days, programmers are also smart enough not to include sensitive data in  
what they send to customers....
```

```
#
```

```
# But not always....
```

```
if input("What's the password? ") == "csawctf{w3_411_star7_5om3wher3}":
```

```
    print("Correct! Congrats! It gets much harder from here.")
```

```
else:
```

```
    print("Trying reading the code...")
```

```
# Notes for beginners:
```

```
#
```

```
# This is Python file. You can read about Python online, but it's a relatively simple  
programming language.
```

```
# You can run this from the terminal using the command `python3 babysfirst.py`, but I'll  
direct you to the internet again
```

```
# for how to use the terminal to accomplish that.
```

```
#
```

```
# Being able to run this file is not required to find the flag.
```

```
#
```

```
# You don't need to know Python to read this code, to guess what it does, or to solve the  
challenge.
```



The flag is embedded in the code itself. If you look closely at the line:



```
python
```

[Copy code](#)

```
if input("What's the password? ") == "csawctf{w3_411_star7_5om3wher3}":
```

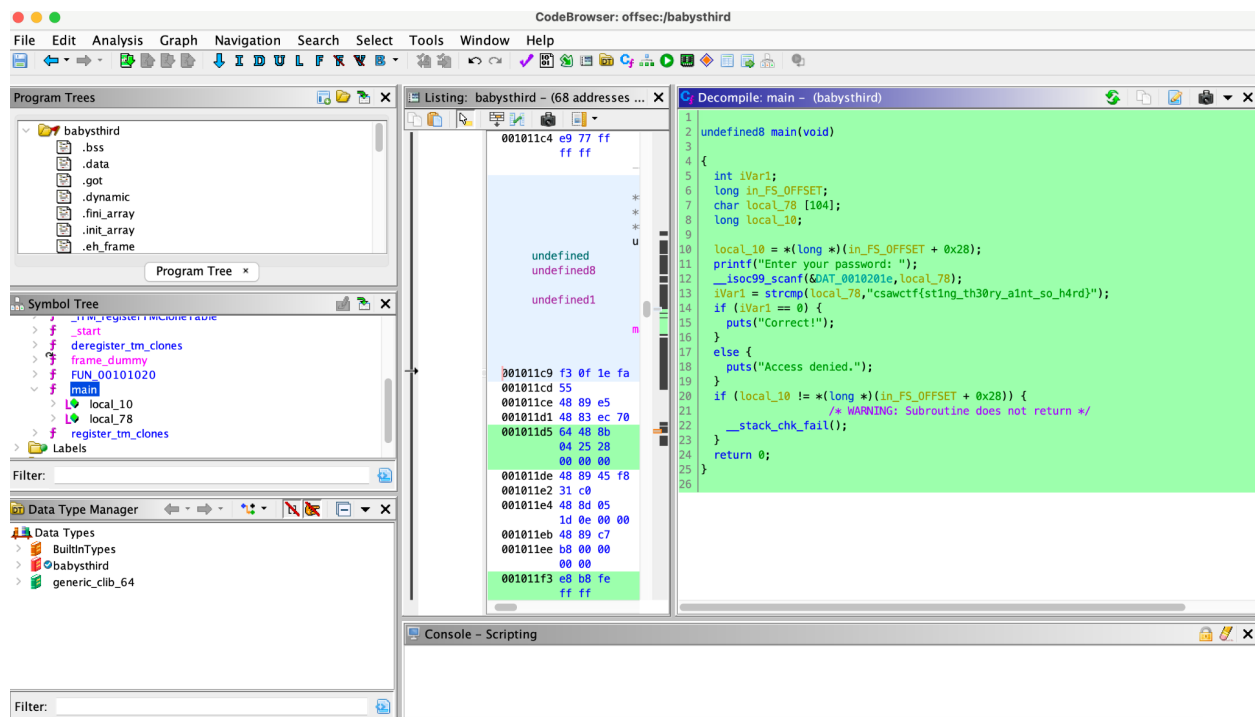
You can see that the correct password is "csawctf{w3_411_star7_5om3wher3}". Therefore,
the flag is:

[Copy code](#)

```
csawctf{w3_411_star7_5om3wher3}
```

Baby's Third

In Baby's Third, after downloading the binary file for baby's third, we used Ghidra to run the binary file and decompile the object code into source code.



Once obtaining the C source code, we used ChatGPT to find the flag using the following prompt. This is a CTF challenge to get the flag from this source code - what is the flag?



This is a CTF challenge to get the flag from this source code - what is the flag?



```
undefined8 main(void)

{
    int iVar1;
    long in_FS_OFFSET;
    char local_78 [104];
    long local_10;

    local_10 = *(long *)(in_FS_OFFSET + 0x28);
    printf("Enter your password: ");
    __isoc99_scanf(&DAT_0010201e,local_78);
    iVar1 = strcmp(local_78,"csawctf{st1ng_th30ry_a1nt_so_h4rd}");
    if (iVar1 == 0) {
        puts("Correct!");
    }
    else {
        puts("Access denied.");
    }
    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
        /* WARNING: Subroutine does not return */
        __stack_chk_fail();
    }
    return 0;
}
```



The flag in the provided source code is:



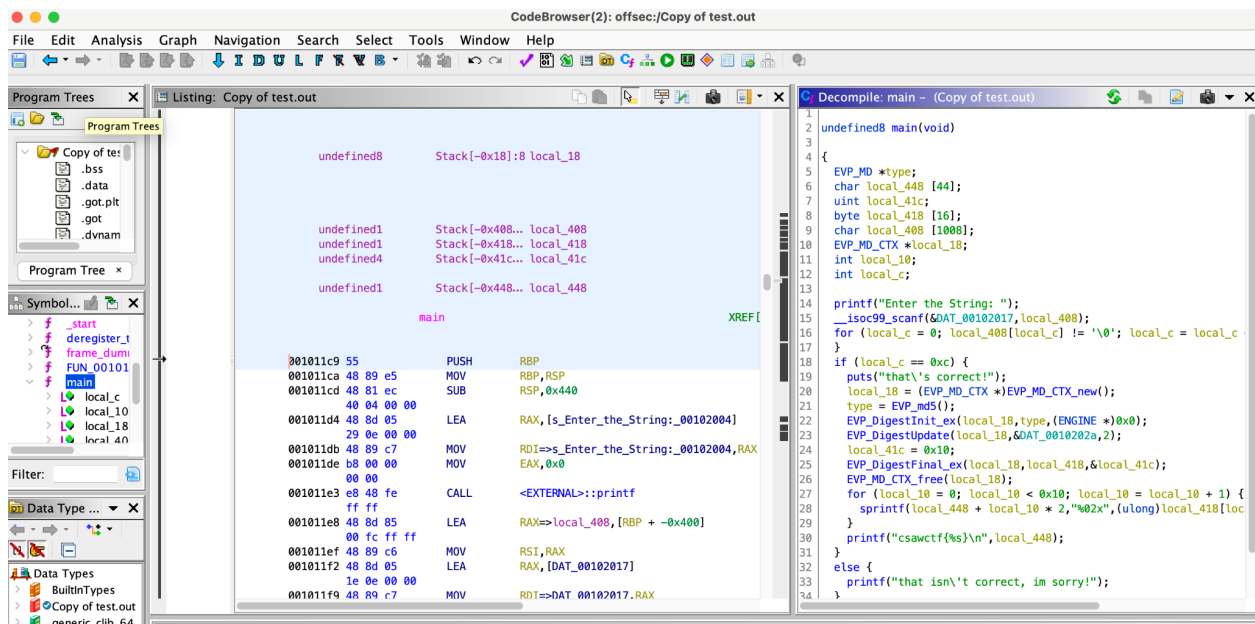
Copy code

```
csawctf{st1ng_th30ry_a1nt_so_h4rd}
```

Flag: **csawctf{st1ng_th30ry_a1nt_so_h4rd}**

Rebug 1

In Rebug 1, after downloading the binary file for baby's third, we used Ghidra to run the binary file and decompile the object code into source code.



Once obtaining the C source code, we used ChatGPT to find the flag using the following prompt. This is a CTF challenge, can you please find the flag?



This is a CTF challenge, can you please find the flag?



```
undefined8 main(void)

{
    EVP_MD *type;
    char local_448 [44];
    uint local_41c;
    byte local_418 [16];
    char local_408 [1008];
    EVP_MD_CTX *local_18;
    int local_10;
    int local_c;

    printf("Enter the String: ");
    __isoc99_scanf(&DAT_00102017,local_408);
    for (local_c = 0; local_408[local_c] != '\0'; local_c = local_c + 1) {
    }
    if (local_c == 0xc) {
        puts("that's correct!");
        local_18 = (EVP_MD_CTX *)EVP_MD_CTX_new();
        type = EVP_md5();
        EVP_DigestInit_ex(local_18,type,(ENGINE *)0x0);
        EVP_DigestUpdate(local_18,&DAT_0010202a,2);
        local_41c = 0x10;
        EVP_DigestFinal_ex(local_18,local_418,&local_41c);
        EVP_MD_CTX_free(local_18);
        for (local_10 = 0; local_10 < 0x10; local_10 = local_10 + 1) {
            sprintf(local_448 + local_10 * 2,"%02x",(ulong)local_418[local_10]);
        }
        printf("csawctf{%s}\n",local_448);
    }
    else {
        printf("that isn't correct, im sorry!");
    }
    return 0;
}
```

 Regenerate



This code appears to be a CTF (Capture The Flag) challenge written in C. It takes user input, performs some operations, and then prints the flag if certain conditions are met. Let's break down the code:



1. The program prompts the user to enter a string.
2. It checks the length of the entered string. If the length is 12 (0xc in hexadecimal), it proceeds; otherwise, it prints an error message and exits.
3. If the length is correct, it initializes an MD5 hashing context (`EVP_MD_CTX`), sets the hashing algorithm to MD5, updates the hash with the constant string `DAT_0010202a` (which is not provided in the code), finalizes the hash, and then converts the hash to a hexadecimal string.
4. The final flag is printed in the format "csawctf{hexadecimal_string}".

Since the constant string `DAT_0010202a` is not provided, you would need to find this value to generate the correct hash and obtain the flag. Without knowing the value of `DAT_0010202a`, it's not possible to determine the exact flag. The constant string might be hidden somewhere else in the challenge or provided separately.

You need to find or infer the missing value to complete the challenge and obtain the flag.

ChatGPT's response indicates that if a string of length 12 is entered by the user, then an md5 hash is converted into a hexadecimal string for the final flag.

```
File Actions Edit View Help
(nyu@kali)-[~]
$ cd Desktop
(nyu@kali)-[~/Desktop]
$ chmod +x test.out
(nyu@kali)-[~/Desktop]
$ ./test.out
Enter the String: abcdefghijkl
that's correct!
csawctf{c20ad4d76fe97759aa27a0c99bfff6710}
```

After entering a string of length 12: 'abcdefghijkl' the flag was retrieved

Flag: csawctf{c20ad4d76fe97759aa27a0c99bfff6710}