**HackRush 2021**

**Start Hacking: HackRushCTF [C]**

Team Name: **CIA**

Shubham Pandit

Dhruv Darda

# We were able to solve the following set of questions:

1. Simple\_check (Reverse Engineering)
2. Ancient (Cryptography)
3. Prime magic 1 (Cryptography)
4. Prime magic 2 (Cryptography)

Note: These questions have been written in the chronological order of solving them.

# 

1. **Simple\_Check (300 points) :**

* **Challenge :** Given a C code, which checks if the given input string is the correct flag, we need to find the flag by analyzing this code.
* **Approach :** In this problem, there were 2 files were given : 1) C program file 2) ELF file of the program
* The C program has been written in a way, so that while giving a string as an input, it declares whether the input string is indeed the flag or not.
* Our task was to find the right flag and submit it.
* I started solving the problem by first understanding ELF files.
* What are ELF files ?
* ELF file stands for the Executable and Linkable Format file.
* The ELF file format is used as a generic object and executable format for binary images used with embedded processors [[1](https://elinux.org/Executable_and_Linkable_Format_(ELF)#:~:text=The%20ELF%20file%20format%20is%20also%20used%20as,by%20file%20data.%20The%20file%20data%20can%20include%3A)]
* In simple terms, this file comprises the binary data produced by the compilers (which compiles certain language program. Here, it is C language), also known as “Machine codes”.
* ELF files can directly be executed in the Linux terminal to run the program (that it has been generated from).
* Though there have been two files given, I didn’t use the ELF file in solving.
* I opened up the C file in the compiler and analyzed the code. The code is given below:

1 #include <stdio.h>

2 #include <stdlib.h>

3 #include <string.h>

4 void fail() {

5 printf("[\*] Wrong flag!\n");

6 exit(0);

7 }

8 void success() {

9 printf("[\*] Correct flag!\n");

10 exit(0);

11 }

12 int main() {

13 printf("I will validate your flag!\n");

14 char flag[50] = "HackRushCTF{";

15 char input[50];

16 printf("Submit your flag: ");

17 scanf("%s", input);

18 if(strlen(input) != 28) {

19 fail();

20 }

22 for(int i = 0; i < 12; i++) {

23 if(flag[i] != input[i]) {

24 fail();

25 }

26 }

27 if(input[12] != 120) {

28 fail();

29 }

30 if(input[13] + input[14] != 110) {

31 fail();

32 }

33 if(input[13] - input[14] != 2) {

34 fail();

35 }

36 if(input[15] != input[21]) {

37 fail();

38 }

39 if(input[16] - input[14] != 48) {

40 fail();

41 }

42 if(input[18] - input[17]\*2 != 16) {

43 fail();

44 }

45 if(input[18] + input[17] != 163) {

46 fail();

47 }

48 if(input[20]\*input[19] != 6148) {

49 fail();

50 }

51 if(input[20] - input[19] != input[12]/2 + 3) {

52 fail();

53 }

54 if(input[21] != 95) {

55 fail();

56 }

57 if(input[24] != 109) {

58 fail();

59 }

60 if(input[26] - input[22] != -53 ) {

61 fail();

62 }

63 if(input[26] + input[22] != 179) {

64 fail();

65 }

66 if(input[23] + 2 != input[25]) {

67 fail();

68 }

69 if(input[23] + input[25] != 100) {

70 fail();

71 }

72 if(input[27] != '}') {

73 fail();

74 }

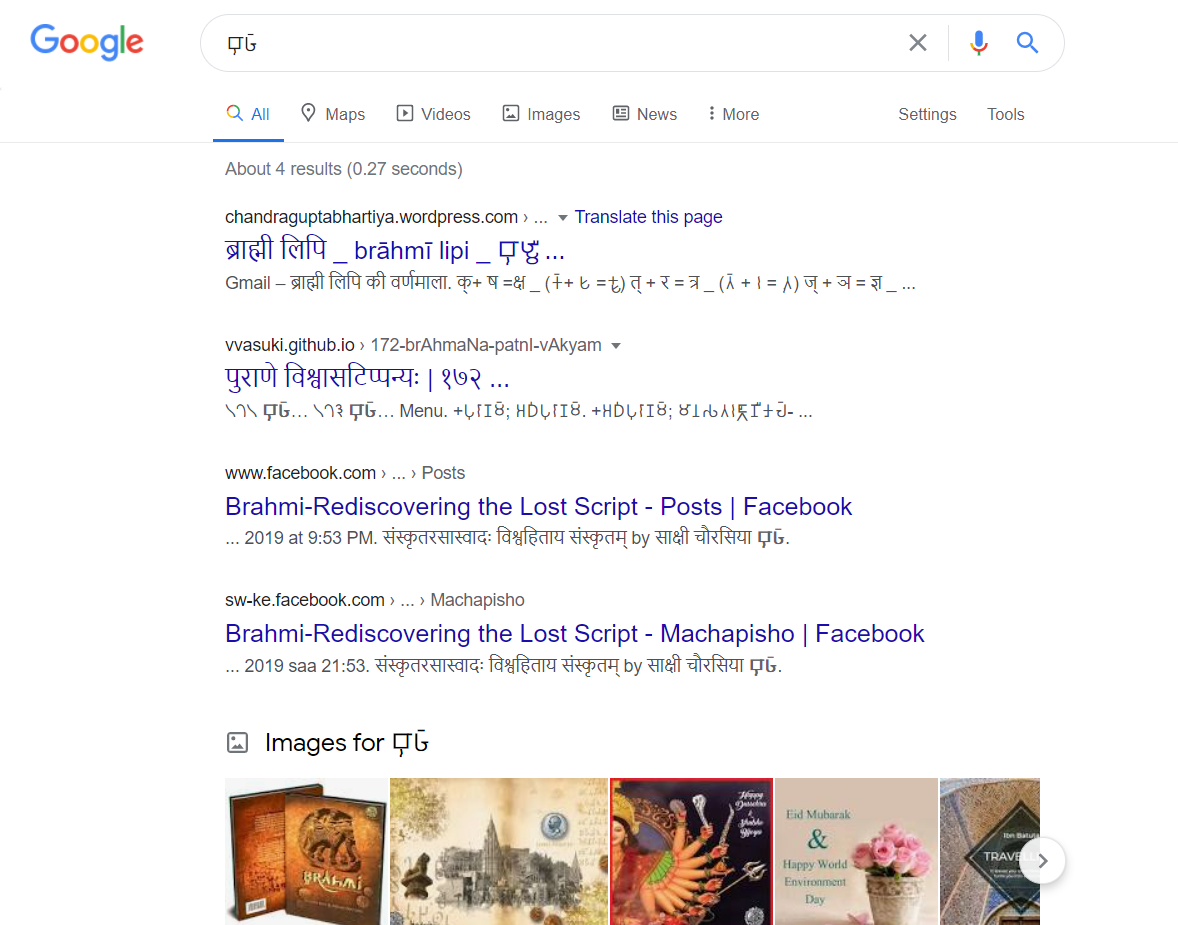
75 success();

76 }

* There are two functions defined here: 1)fail 2) success
* “Fail” is being called whenever any condition, mentioned in the \_\_main\_\_, is dissatisfied and if all the conditions are satisfied or in other words, all conditions are passed, then it calls “success” function and prints that the input string (given as flag) is the correct flag.
* I reverse engineered all these conditions and found the flag.

1. **Ancient (50 points) :**

* **Challenge**: found some weird text, Can you find out what this means?
* 𑀩𑁆𑀭𑀸𑀳𑁆𑀫𑀻 𑀮𑀺𑀧𑀺 𑀪𑀸𑀭𑀢 𑀓𑀻 𑀧𑁆𑀭𑀸𑀘𑀻𑀦𑀢𑀫 𑀮𑀺𑀧𑀺𑀬𑁄ं 𑀫𑁂ं 𑀲𑁂 𑀏𑀓 𑀳𑁃 𑀬𑀳 𑀳𑁃 𑀆𑀧𑀓𑀸 𑀚𑀯𑀸𑀩 : HackRushCTF{𑀅𑀰𑁄𑀓​}
* **Approach** : I searched the first letter of these cipher, which is 𑀩𑁆𑀭𑀸. I found the following result:



* It is evident from the above, that the given symbols are from “Brahmi Script”, Then, using this [converter](https://aksharamukha.appspot.com/converter), I translated it into “Gujarati” and the translation is :

“બ્રાહ્મી લિપિ ભારત કી પ્રાચીનતમ લિપિયોं મેं સે એક હૈ યહ હૈ આપકા જવાબ : HackRushCTF{અશોક​}”

* Converting into English, it is:

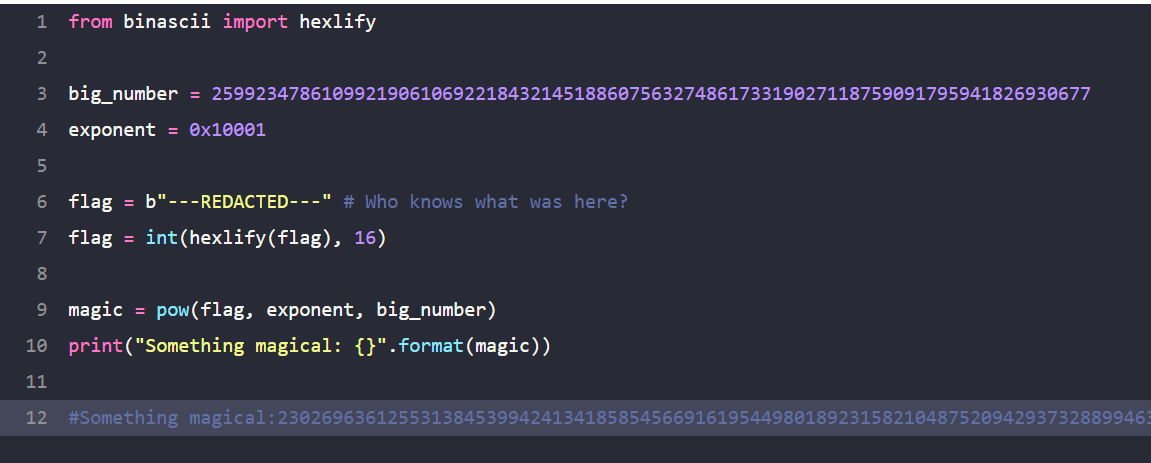
“Brahmi script is one of the oldest scripts in India. Your answer:

HackRushCTF{ashok}”

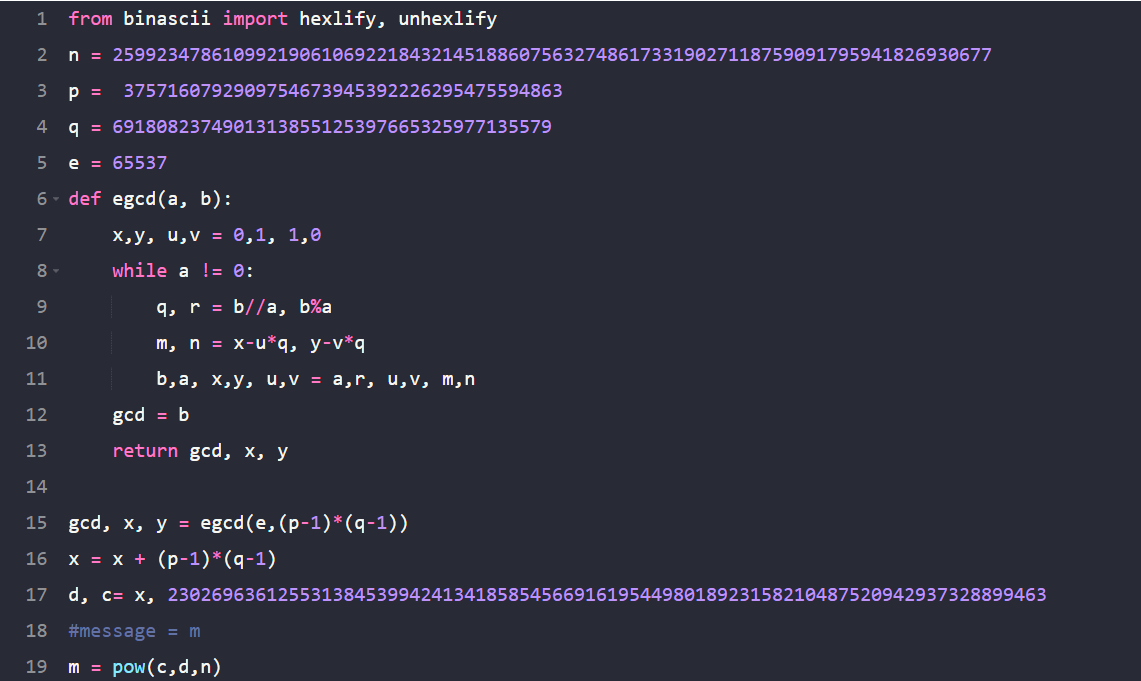
* Thus, the flag is found. It is: HackRushCTF{ashok}

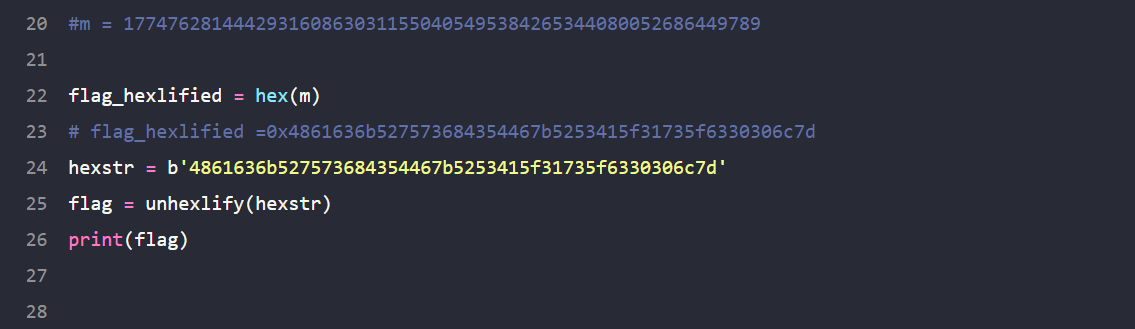
1. **Prime magic 1 (100 points):**

* **Challenge :** A program is given, written in Python. We need to find the flag hidden inside it.
* **Approach :** The code is given below:



* This code uses RSA encryption to encrypt the flag, where the public key (n,e) is given by (big\_number, exponent).
* First, it encrypts the flag using the public key, i.e.
* encrypted\_flag = (flag)\*\*e mod n
* Before encrypting the flag, it converts the original\_flag into hexadecimal bytes, using the **hexlify()** function from the **binascii** module.
* **int()** function is used to convert the hexadecimal number into decimal one.
* Now, the encrypted\_flag is stored in variable “**magic**”.
* We are given the output of the program.
* **Solution :** I wrote the following code to decrypt the RSA encryption :



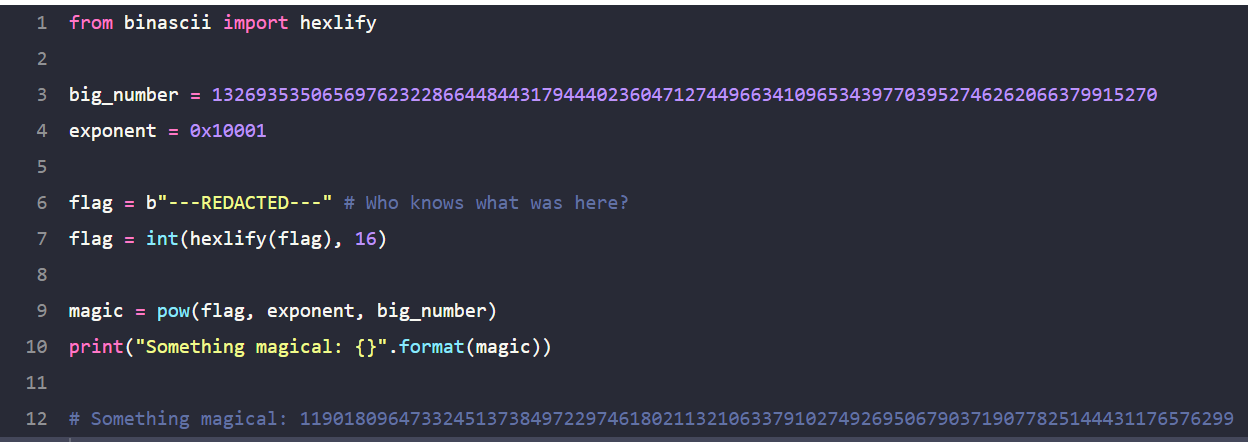


# Some functions of this code were taken from [this site](https://stackoverflow.com/questions/49856115/inverse-of-a-powa-b-n-function-in-python-decryption-code).

* For decrypting RSA code, we need the private key (n, d) and the ciphered message.
* Then, it could be deciphered by :
* **original\_messeage = (CipherText \*\* d) mod n**
* This method requires us to factorize **n**  into two **prime factors : p** & **q**.
* Then, we can easily proceed to find the flag.
* **Answer** : **HackRushCTF{RSA\_1s\_c00l}**

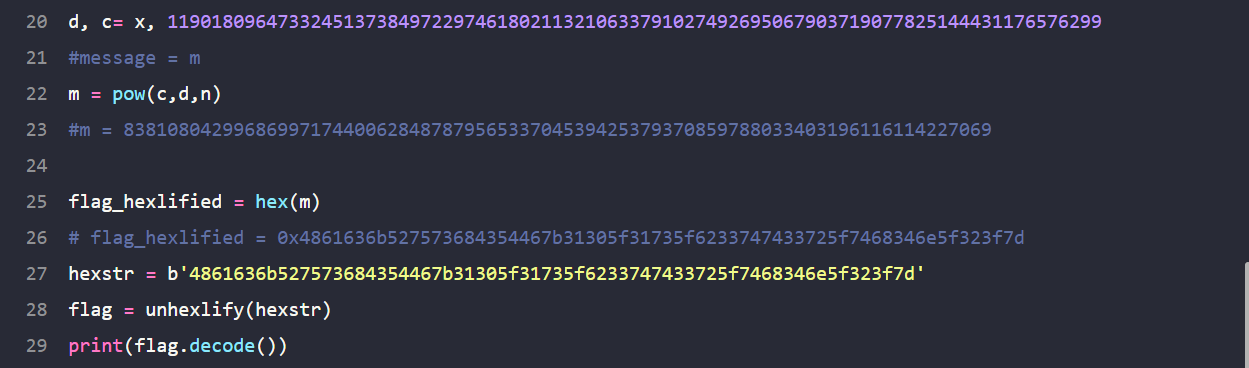
1. **Prime magic 2 (300 points):**

* **Challenge :** This problem is similar to the previous problem. There is a code given, which encrypts the original flag using RSa encryption and we need to decrypt it and find the original flag.
* **Approach :** The code is given below:



* The program is very similar to the previous problem. In fact, there is **just one change**  in this code : the **big\_number**  is changed.
* To be precise, in this program:
* **big\_number = 510510 \*(** previous\_**big\_number)**
* Therefore, this is **NOT** the ordinary RSA encryption, but **Multi-prime RSA Encryption**.
* To solve it, we need to factorize the **big\_number** into all possible **prime factors**.
* Then, we can use the same **Extended Euclidean Algorithm** to find the **private key** (which is given by, **(n, d)**) .
* **Solution :** It uses the exact same functions, used in the previous problem. The code is given below:





* **Answer : HackRushCTF{10\_1s\_b3tt3r\_th4n\_2?}**