Uday Ojha PicoCTF writeup-  
Shop- After connecting to the server using netcat, we are greeted with the shop itself, so in one of the hints it says to always check edge cases, so what i did was that instead of buying a normal amount of apples i just bought negative amount, this caused the system to give me alot of money instead of taking it, now after this was done i got enough money to be able to afford the Fruitful flag item which was previously too expensive and then once you buy it, it gives you the flag in the form of decimal values what i did was put that in rapid tables website and it gave me the flag.

Hurry Up, wait!-

What i did in this challenge was that first i downloaded the svchost.exe, and then i opened it with ghidra, after looking around for a long time i found a class that resembled something half understandable. Now inside this class there were multiple functions each that printed one specific character, now to find out what character each function printed was as simple as clicking on each function and then it took you to another page that explained the function and then you clicked on it again and then it took you to a page which showed which hex value corresponded to the letter being printed, Like this i looked at each of the function and after reading all of the functions you get the flag.

Vault Door 1- First i downloaded the java file that they provided, i nano'd the java file and in the code i saw that the password was provided it just wasn't sorted so what i did was just

I took all those password.charAt(?) lines and pasted them into a new file, which I used sort -V on that gave me all the digits of the password which was sorted. Put that into the flag on pico.

Vault Door 3- In this source code we see multiple loops and the input statement that should go into those loops that will rearrange the strings and give the flag, so what i did was make a new java file and copy pasted all loops into it and modified the code so that it would take that input given to us and print me the output, the java code i made is-

"

class VaultDoor3sol{

public static void main (String args[]){

char[] buffer = new char[32];

String password = "jU5t\_a\_sna\_3lpm18gb41\_u\_4\_mfr340";

int i;

System.out.println("This is a test");

for (i=0; i<8; i++) {

buffer[i] = password.charAt(i);

}

for (; i<16; i++) {

buffer[i] = password.charAt(23-i);

}

for (; i<32; i+=2) {

buffer[i] = password.charAt(46-i);

}

for (i=31; i>=17; i-=2) {

buffer[i] = password.charAt(i);

}

System.out.println(buffer);

}

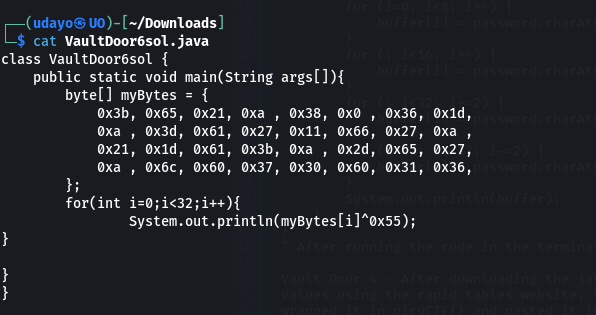
}

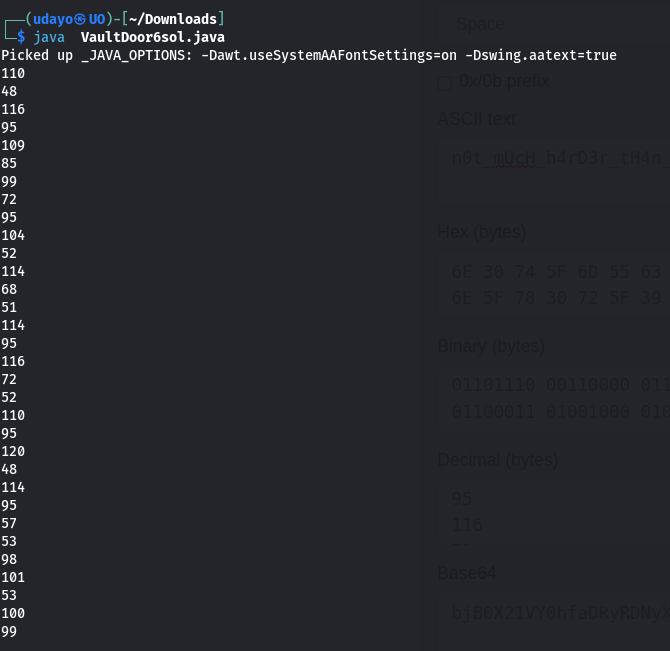
" After running the code in the terminal through the java command it printed out the flag.

Vault Door 4 - After downloading the java file given to us i opened it using nano, after looking at the code what i did was that i just converted the given Bytes to us in the myBytes[]= into Ascii values using the rapid tables website, the first row was decimals , second row was hexadecimal values and the third was in the octal format, which after converting in each row individually i just wrapped it in picoCTF{} and pasted it into the site.

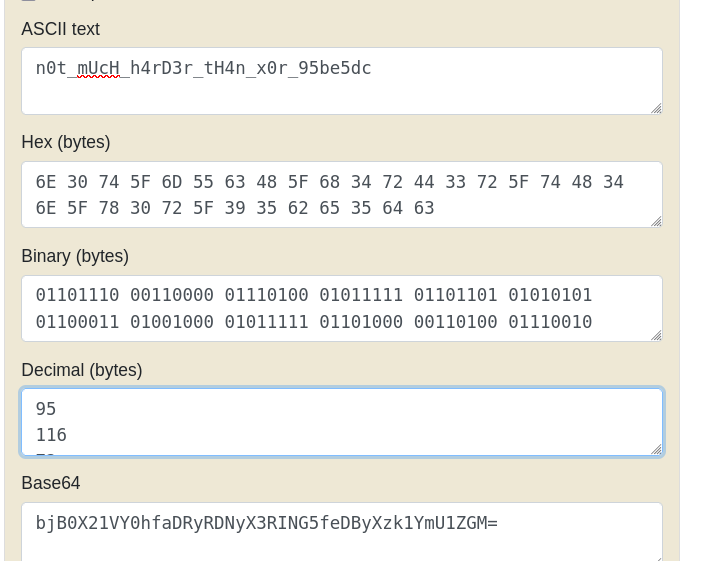
Vault Door 5 - First i copy pasted the strings given in the source code at the bottom and put it into an online base64 decoder which gave me a bunch of hex bytes which I passed through the same rapid tables website and converted it into ascii keys giving me the flag.

Vault Door 6- So what we had to do in this challenge was that after reading the source code provided to us you would understand that each of the hexadecimal values being passed were being xor with the static value of 0x55 and then being minused with the stored value to see if we get zero, this implied that if we just xorred the given hex values with 0x55 we would get the flag, so what i did was that i made a new java file, and wrote the following code in it.



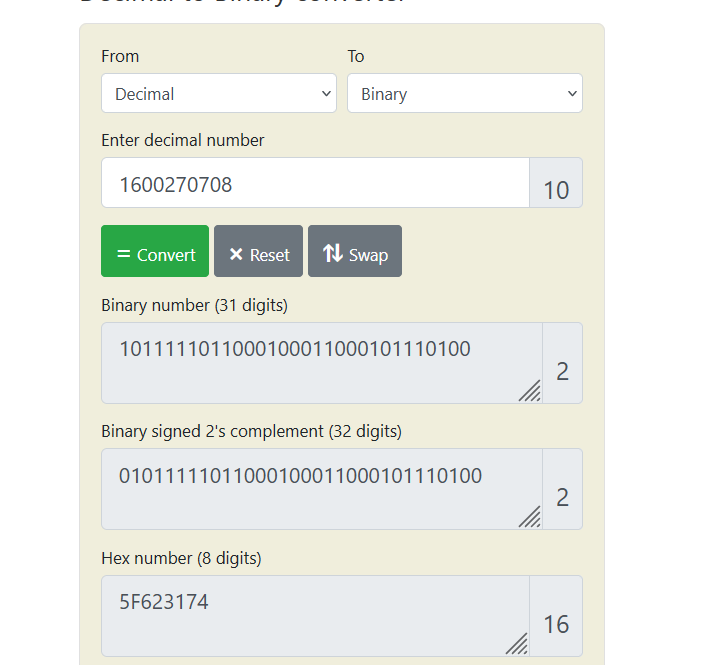
After running this code you get the decimal values, 

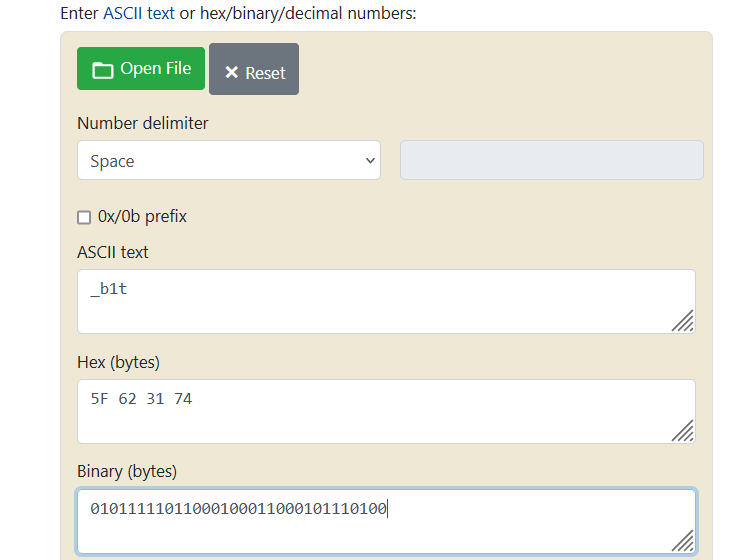
You can just put these in rapid table and convert it into ascii giving you the flag.



Vault Door 7-

What i did in this challenge was that i took all the decimal values that they gave us i converted them into a binary number and then that binary number i converted into a ascii text using rapid tables.

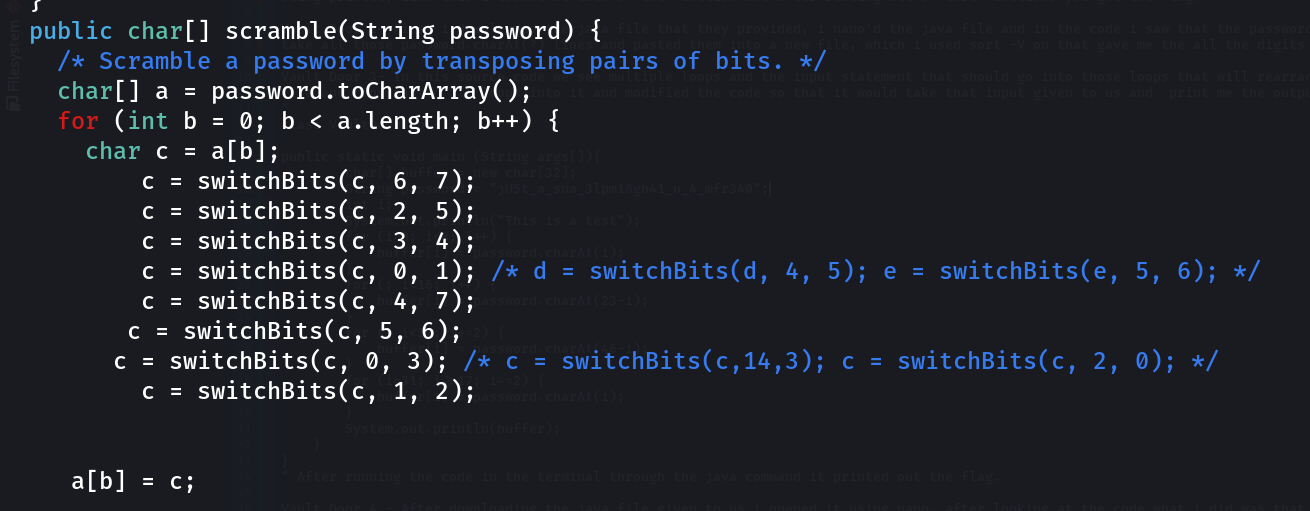


  
Doing this gave me the flag needed.

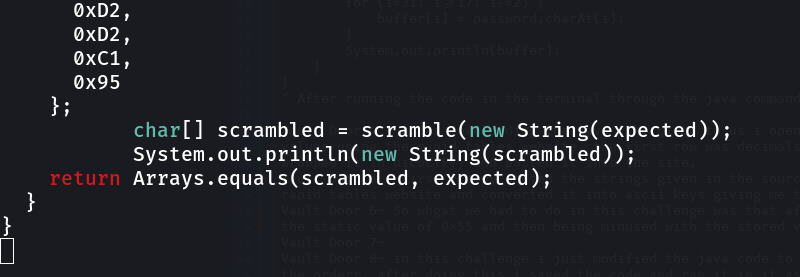
Vault Door 8- in this challenge i just modified the java code to instead of scrambling the inputted password it just scrambles the expected answer and the order of the scramblin i manually reversed the order, after doing this i saved the code and ran it in it asked for the password, you have to at least give the following phrase "picoCTF{}" for it to accept your input after doing all this the password is printed regardless of what you enter inside those curly braces



The edited code is-



//Upside down reversed order of switch bits



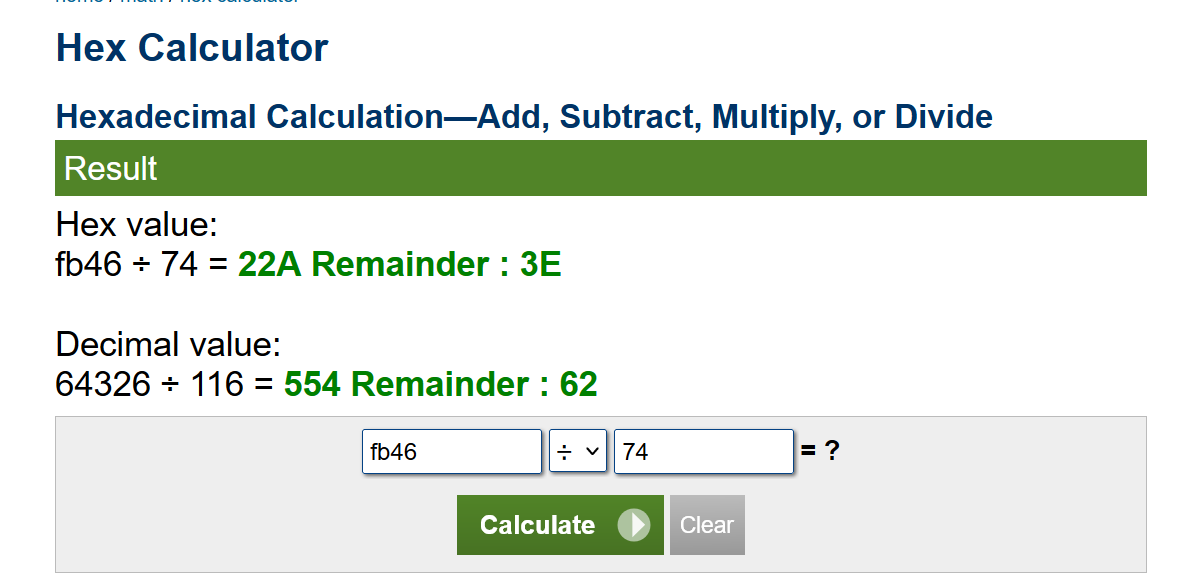
//printing the pass.

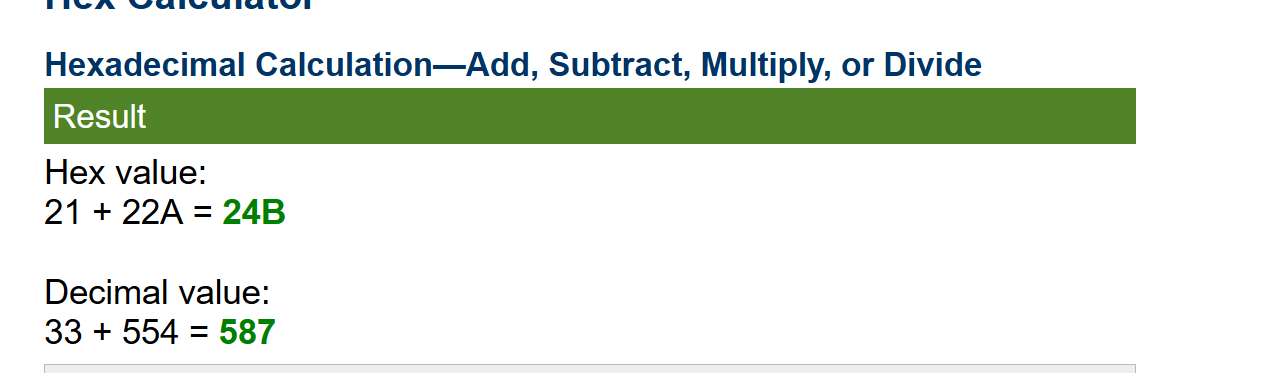
Asm1-

we are putting whatever hex value we get given in the question (0x6fa)into the stack. This value gets pushed into ebp and then moved into esp on lines 0 and 1. We have to follow the code carefully until in one of the steps it asks you to add a specific value to your the inputted hex value, which later it just pops, in the question it tells us that the flag this time around won't have the standard picoCTF{} notation so i just put that hex value into the website and that gives me the correct answer.

Asm2-

With the help of Martin Carlisle i understood vaguely what was going on

These are the asm2(0x4,0x21),we know that ebp-0x4 is storing 0x21 and ebp-0x8 is storing 0x4. We then take an unconditional jump to line 31.We see here that we are comparing the value stored at ebp-0x8, which is 0x4, to 0x47a6. Since the comparison is less or equal to and the condition is jle (jump if less then or equal), we make the jump back up to line 20, basically a For loop forms over here.The value at ebp-0x4 rises by 0x1, and ebp-0x8 rises by 0xa9. The jle condition causes this to keep looping until ebp-0x8 is not less than or equal to 0xfb46.After doing the math considering how many times it will it loop and how many times you have to add we get the value.

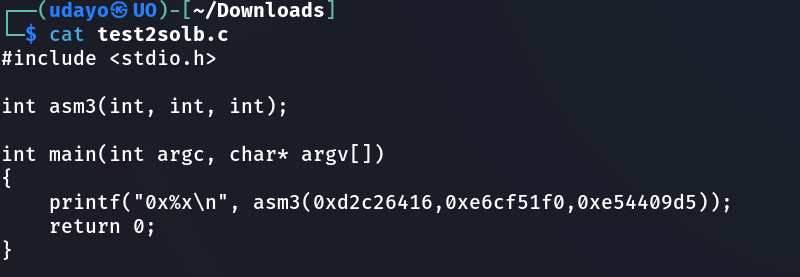


Since it needs to be more than we add one more making it 24C making the flag,

0x24C.

Asm3 & 4- For both of these challenges I did a similar thing in the sense that I compiled the assembly code in C and ran it with the values they provided as the function parameters using gcc. They both have similar steps so I am taking the liberty of combining them both. Essentially what you have to do is make a couple of changes to the assembly code and make a new C program that basically calls the assembly code as a function. In the assembly code you have to

make the following changes.

// you can make a new C file like the one i made below and give a new function with the same name as assembly code. And then print the output of the of the function after calling it in main, giving the function parameters as what was given in the picoCTF website.

And in the assembly source code site you have to make the following changes.

.intel\_syntax noprefix

.global asm3/asm4  
  
What these lines do is act as an assembler directives and do not represent actual assembly instructions that perform a specific operation on the CPU.

The first directive, ".intel\_syntax noprefix", sets the syntax mode of the assembler to use Intel syntax instead of the default AT&T syntax. This means that the assembler will interpret the subsequent assembly code using the Intel syntax rules.

The second directive .global asm3, is used to declare a global label named asm3. This means that the label can be accessed from other parts of the program, including from other source files. Global labels are used to define entry points into the code or to provide access to data or functions that are used across different parts of the program.

Now after you have made these files what we do next is make use of a software called gcc which is basically a C compiler, and we give it these files as input and it gives us a runnable program that will give us print us the flag, I looked up these commands online basically what we have to put in the terminal is-   
 gcc -masm=intel -m32 -c filename.S -o filename1.o

gcc -m32 -c filename.c -o filename2o

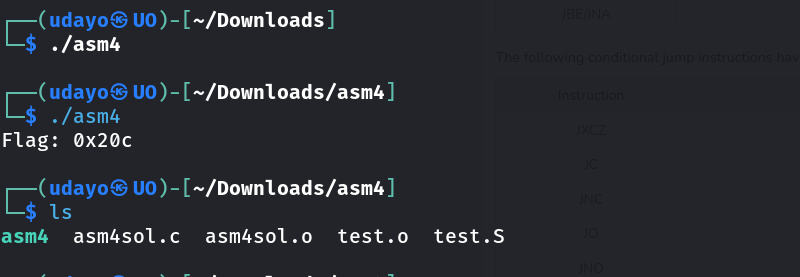
gcc -m32 filename1.o filename2.o -o filename3

// the -c is just for the input file and the -o is for where it has to be outputted to.

// -m32 tells the gcc to generate the code in 32 bits.

// -masm=intel tells the program to make use of the intel architecture.

This will output an executable which you can run to get the flag-

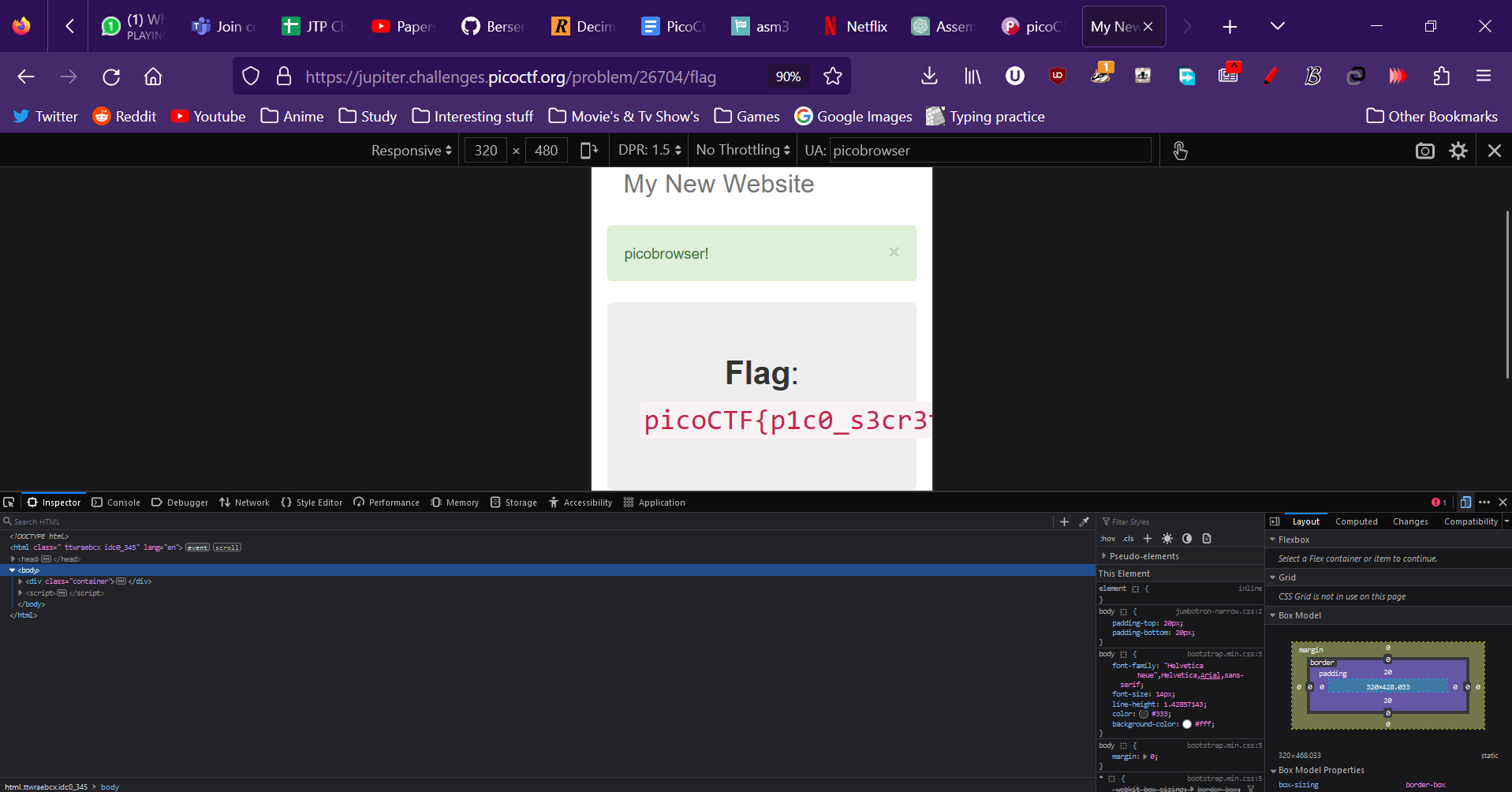


Web-

logon-

In this specific challenge what i did was that in the website i checked out the inspect element portion of the website, in that i went to storage and in the storage there were multiple tabs i checked out all of them in which the cookies gave me something to look at, now we have the ability to alter these cookies there was one field that mentonied admin -> False, so what i did was change the 'False' to "True", and then i reloaded the website giving me access to the website.

Pico Browser- In this specific task what i did was first i went to the website and then again checked out the inspect element location, over there is a button called Responsive design mode, in this specific mode you can change your user agent, in that UA- tab i just typed picobrowser and that made thee website think i'm accessing it from a browser called picobrowser and it displayed the flag.



WebGauntlet-

level 1 & 2 & 3- i just put "admin';" doing this stops the sql statement at only the username stage giving us access to the next level this works for both level 1 and 2.

level 5 & 4 - since the word admin gets filtered out now what i decided to do was that i split the word admin into "adm" and "in".so in the username prompt area i put the following, "adm'||/\*" and in the password prompt area i put "\*/'in", this way everything in between gets commented out and the word admin gets passed through and lets us clear the level for both 4 and 5 as they both dont have a filter against it.

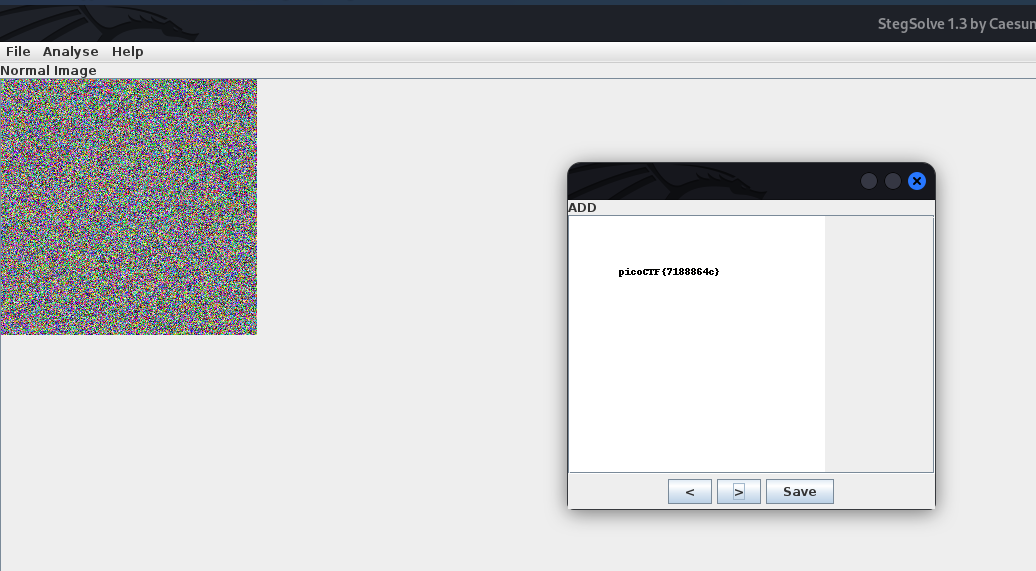
A helpful source I learned from for this level was by Martin Carlisle on youtube.

Cryptography-

IIn

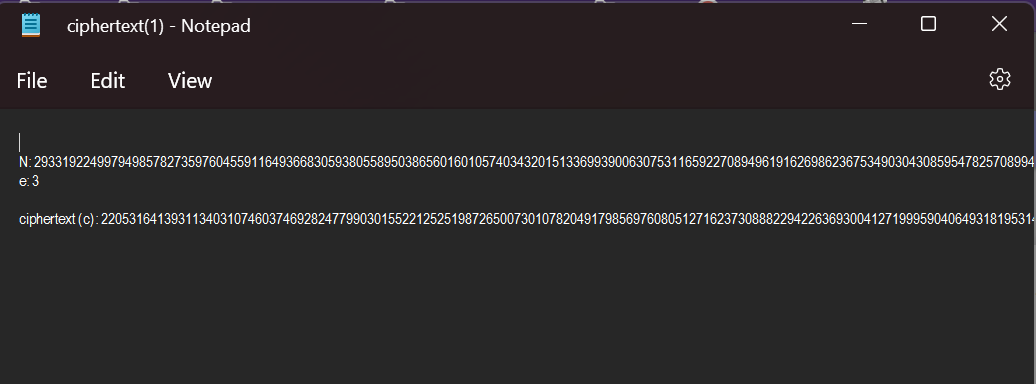
Pixelated-

In this challenge what we were provided with were two scrambled images and we had to combine both the images in such a way that it would give us the flag, so what i did was i used this tool call StegSolve and inputted both the images into that and it gave me the flag. What this tool did was take both the images and it took each individual bytes of both the images and it will add it together.



MiniRSA-

In this challenge they provided the data required for decryption so what i did was just pass it through a website called dcode.fr.





Dachshund Attack-

In this challenge I did the same as above. I just connected to the server using netcat got the details, put it into the dcode rsa cipher and I got the flag.