My ctf is to guess the password, because guessing the password itself is very simple, so I want to talk about my original ctf design first. I originally wanted to write a finger-guessing game. The server sends options to the client, and the player needs to defeat the client 10 times in a row to get the flag. In this case the player needs to catch the client's actions by listening for data exchange to defeat it. But by the time I considered it, we had learned and tried more advanced attacks using tools like Wireshark. In order to avoid the same stuff, I finally gave up this idea. But I think this game would be funny, so I want to share it with you.

Then I made this ctf about guessing passwords. Although it looks very simple. Actually, it is pretty simple. But if you want the fastest way to unlock the password, it would take some work. This password itself is a 4-digit password, as long as the correct password is entered, you will get the flag. But the server sets a time limit. If the time is exceeded, it will automatically disconnect. Then a new password will be regenerated on the next connection. But the server will give the user some hints after each password entry as compensation.

So the essence of this CTF is to let the test taker find the correct answer as fast as possible when you can know the correct rate of your passwords. If you use brute force without any hint provided by the server, a 4-digit password requires 4 nested loops to guess, and the worst result may be 62 to the 4th power times to guess the correct password. 62 formed by 10 regular numbers from 0 to 9, and 26 letters with its uppercase. The complexity of this brute force is n to the 4th power. Which is bad. But if you can use the hint, you can use 4 loops in sequence and get the answer like the right part. In the worst case, it only takes 4*62 guesses to find the correct answer. In my design, such brute force speed is acceptable. So this CTF is very simple to do. But this is not the fastest solution. If you want to find a faster solution, it will take your time.

I have an idea that only needs to loop 0-9 and ascii letters once to find the correct password. With these hints, we can iterate all possible characters and record the correctness. Four letters as a group. Once we detect a correct letter in the loop, we can try a verification step to find out the correct character and its correct position. In one verification step, we need to try one character in four places, for 4 characters, we will try it 16 times. For four correct digits, we will try verification steps 4 times. Finally, we only loop once with additional 16 * 4, which is 64 times and get the correct password. The total iteration steps in the worst case would be 62 / 4 + 64 = 80. This is where my ctf encourages everyone to try. If you are interested, you can also think about another faster algorithm yourself.

Thanks for listening.