PW Crack 4

PicoCTF

Fares Zuleta

February 21, 2026

Contents

1 Description

Can you crack the password to get the flag? Download the password checker and you'll need the encrypted flag and the hash in the same directory too. There are 100 potential passwords with only one being correct. You can find these by examining the password checker script.

2 Files

After downloading the files, we can see three files in our directory.

```
[Zer0th@Arch PWCrack4]$ ls
level4.flag.txt.enc level4.hash.bin level4.py
```

When reading the script, we can observe the following code.

3 Script

```
import hashlib

### THIS FUNCTION WILL NOT HELP YOU FIND THE FLAG --LT
    ########################
def str_xor(secret, key):
    #extend key to secret length
    new_key = key
    i = 0
    while len(new_key) < len(secret):
        new_key = new_key + key[i]
        i = (i + 1) % len(key)
    return "".join([chr(ord(secret_c) ^ ord(new_key_c)) for (secret_c,new_key_c
        ) in zip(secret,new_key)])
###############################################################################


flag_enc = open('level4.flag.txt.enc', 'rb').read()
correct_pw_hash = open('level4.hash.bin', 'rb').read()


def hash_pw(pw_str):
    pw_bytes = bytearray()
    pw_bytes.extend(pw_str.encode())
    m = hashlib.md5()
    m.update(pw_bytes)
    return m.digest()


def level_4_pw_check():
    user_pw = input("Please enter correct password for flag: ")
    user_pw_hash = hash_pw(user_pw)
    if( user_pw_hash == correct_pw_hash ):
        print("Welcome back... your flag, user:")
        decryption = str_xor(flag_enc.decode(), user_pw)
        print(decryption)
        return
    print("That password is incorrect")



level_4_pw_check()



# The strings below are 100 possibilities for the correct password.
# (Only 1 is correct)
pos_pw_list = ["6288", "6152", "4c7a", "b722", "9a6e", "6717", "4389", "1a28",
     "37ac", "de4f", "eb28", "351b", "3d58", "948b", "231b", "973a", "a087",
    "384a", "6d3c", "9065", "725c", "fd60", "4d4f", "6a60", "7213", "93e6", "8
    c54", "537d", "a1da", "c718", "9de8", "ebe3", "f1c5", "a0bf", "ccab",
    "4938", "8f97", "3327", "8029", "41f2", "a04f", "c7f9", "b453", "90a5", "25
    dc", "26b0", "cb42", "de89", "2451", "1dd3", "7f2c", "8919", "f3a9", "b88f
    ", "eaa8", "776a", "6236", "98f5", "492b", "507d", "18e8", "cfb5", "76fd",
    "6017", "30de", "bbae", "354e", "4013", "3153", "e9cc", "cba9", "25ea", "
    c06c", "a166", "faf1", "2264", "2179", "cf30", "4b47", "3446", "b213", "88
    a3", "6253", "db88", "c38c", "a48c", "3e4f", "7208", "9dcb", "fc77", "e2cf
    ", "8552", "f6f8", "7079", "42ef", "391e", "8a6d", "2154", "d964", "49ec"]
```

4 Analysis

After analyzing the source code, we can extract 100 passwords from it. We now need to create our script to brute-force the password.

5 Creating the script

Instead of manually testing each candidate, we can create a script to automate the hash comparison process. The challenge already provided us with a .hash.bin file.

```
level4.hash.bin
```

To read the file we have to execute the following command in our Linux system.

```
hexedit level4.hash.bin
```

5.1 Hexedit

Then it displays the hexadecimal values of the hash key (MD5).

```
1C 92 41 5F 2F C0 8B 0E 8A 0E BB 6F 3F 21 CD CC
```

5.2 Python

After reading the files we already have all the information to create our own script.

```
import hashlib

PW = ["6288", "6152", "4c7a", "b722", "9a6e", "6717", "4389", "1a28", "37ac",
    "de4f", "eb28", "351b", "3d58", "948b", "231b", "973a", "a087", "384a", "6
    d3c", "9065", "725c", "fd60", "4d4f", "6a60", "7213", "93e6", "8c54", "537d
    ", "a1da", "c718", "9de8", "ebe3", "f1c5", "a0bf", "ccab", "4938", "8f97",
    "3327", "8029", "41f2", "a04f", "c7f9", "b453", "90a5", "25dc", "26b0", "
    cb42", "de89", "2451", "1dd3", "7f2c", "8919", "f3a9", "b88f", "eaa8", "776
    a", "6236", "98f5", "492b", "507d", "18e8", "cfb5", "76fd", "6017", "30de",
     "bbae", "354e", "4013", "3153", "e9cc", "cba9", "25ea", "c06c", "a166", "
    faf1", "2264", "2179", "cf30", "4b47", "3446", "b213", "88a3", "6253", "
    db88", "c38c", "a48c", "3e4f", "7208", "9dcb", "fc77", "e2cf", "8552", "
    f6f8", "7079", "42ef", "391e", "8a6d", "2154", "d964", "49ec"]
key = "1c92415f2fc08b0e8a0ebb6f3f21cdcc"

for pw in PW:
        hashed = hashlib.md5(pw.encode()).hexdigest()
        if hashed == key:
           print(pw)
           break
```

5.3 Explaining our script

The following line imports the hashlib module to compute MD5 hashes. Then we define the password list containing all candidate passwords. The key line stores the hexadecimal representation of the key we previously obtained. The hexadecimal hash was converted to lowercase to match the format returned by hexdigest().

```
import hashlib

PW = ["6288", "6152", "4c7a", "b722", "9a6e", "6717", "4389", "1a28", "37ac",
    "de4f", "eb28", "351b", "3d58", "948b", "231b", "973a", "a087", "384a", "6
    d3c", "9065", "725c", "fd60", "4d4f", "6a60", "7213", "93e6", "8c54", "537d
    ", "a1da", "c718", "9de8", "ebe3", "f1c5", "a0bf", "ccab", "4938", "8f97",
    "3327", "8029", "41f2", "a04f", "c7f9", "b453", "90a5", "25dc", "26b0", "
    cb42", "de89", "2451", "1dd3", "7f2c", "8919", "f3a9", "b88f", "eaa8", "776
    a", "6236", "98f5", "492b", "507d", "18e8", "cfb5", "76fd", "6017", "30de",
     "bbae", "354e", "4013", "3153", "e9cc", "cba9", "25ea", "c06c", "a166", "
    faf1", "2264", "2179", "cf30", "4b47", "3446", "b213", "88a3", "6253", "
    db88", "c38c", "a48c", "3e4f", "7208", "9dcb", "fc77", "e2cf", "8552", "
    f6f8", "7079", "42ef", "391e", "8a6d", "2154", "d964", "49ec"]
key = "1c92415f2fc08b0e8a0ebb6f3f21cdcc"
```

The `for` loop iterates through each candidate password, computes its MD5 hash, and compares the result with the extracted hash value.

```
for pw in PW:
       hashed = hashlib.md5(pw.encode()).hexdigest()
       if hashed == key:
          print(pw)
          break
```

6 Executing our script

We now execute the script to determine the correct password without manually guessing.

```
[Zer0th@Arch PWCrack4]$ python3 unhash.py
973a
```

As shown above, the password obtained is ”973a”.

7 Executing the challenge’s script

After successfully obtaining the password, we test it in the level4.py script to verify its correctness.

```
[Zer0th@Arch PWCrack4]$ python3 level4.py
Please enter correct password for flag: 973a
Welcome back... your flag, user:
picoCTF{fl45h_5pr1ng1ng_ae0fb77c}
```

The flag was successfully retrieved.