How to Decrypt MD5 Passwords in Python?

infosecscout.com/decrypt-md5-python

Patrick Fromaget

The big question many beginners have about MD5 is how to decrypt hashes after encryption.
In this post, I'll explain you this, and specifically, how to do this in Python.

How to Decrypt MD5 Passwords in Python?
The MD5 cryptographic algorithm is not reversible.
A word can be encrypted into MD5, but it's not possible to create the reverse function to decrypt a MD5 hash to the plain text.
To validate MD5 passwords in Python, there is a different solution.

In this tutorial, I'll start by a brief introduction about the MD5 algorithm.
Then I'll show you how to validate passwords in Python, without any need to decrypt the hash.
And I will finish this post by my solution to try decrypting MD5 hashes in Python, if it's really your goal today.

By the way, if you are interested in how MD5 decryption really works, I highly encourage you to take a look at my e-book "The Secrets of MD5 Decryption"here. It explains everything you need to know, going directly to the point with practical examples you can

test on your computer. You don't need any hardware to get started, just a few tips I give in this book.

Master Linux Commands
Your essential Linux handbook
Want to level up your Linux skills? Here is the perfect solution to become efficient on Linux. **20% off today!**

Download now

Reminder about the MD5 algorithm

I often start with an introduction about the MD5 algorithm on this blog because most people have difficulties to understand the logic behind the MD5 algorithm.

MD5 encryption

Hide your IP address and location with a free VPN:
Try it for free now, with advanced security features.
2900+ servers in 65 countries. It's free. Forever.
So, **MD5 is a cryptographic algorithm that generate a string with 32 hexadecimal characters, whatever the word or text length you try to encrypt.**
Even an ISO file from several gigabytes can be hashed in 32 characters with the MD5 algorithm.

Pseudo-code example:

```
MD5("MD5Online") = d49019c7a78cdaac54250ac56d0eda8a
```

This information provides the following algorithm results:

- **The output is always 32 characters long** but you can hash anything in 32 characters.
- So, the MD5 algorithm output is not unique. Two word or files can have the same MD5 hash.
- Given this information, it's not possible to reverse a hash to the original word.

If you are interested to understand all the details, I recommend reading this page (Wikipedia) or picking a course or book from my resource page.

MD5 decryption

So, why the MD5 algorithm is so fascinating if decrypting hashes is not possible?
The MD5 algorithm has a weakness we can exploit, each time you create a MD5 hash of a word, you get the same result.
As this algorithm was the principal one in the world a few decades ago, many databases exists with the corresponding word for each MD5 they know.

So, there is no decryption algorithm for MD5, but there is a solution.
For example, you now know that the MD5 hash from “MD5Online” is d49019c7a78cdaac54250ac56d0eda8a.
If someone is looking for the word corresponding to this hash, there is a good chance that “MD5Online” was the original password.

That’s what is used for MD5 decryption in general.
And especially **on MD5Online.org, we have a huge database with over a trillion hashes stored inside**. You can access this database with our tools.
There are other solutions, but it’s the main one.

MD5 Passwords validation with in Python

Theory (pseudo-code)

In any language, the MD5 functions are really fast to encrypt a password.
So you can use it in your application without any performance issue.

That’s the reason why some developers are using the MD5 algorithm to hide passwords in their database.
To verify the login credentials, they just encrypt the typed password in MD5 and compare this hash to the one stored in database.
If there is a match, we consider that the login is valid (even if the encryption is not unique, it’s not a big deal).

The pseudo-code can look like this:

```
IF (MD5(PASSWORD_ATTEMPT) == DATABASE_PASSWORD)
THEN LOGIN_SUCCESS();
ELSE LOGIN_ERROR();
```

We’ll now see how to do this in Python specifically.

Python examples

Encrypting password in Python

So in Python, there is a library available directly to manage MD5 hashes, it’s “hashlib”.
For information, this library can handle many other algorithms like the SHA variants (SHA1, SHA256, SHA512, …) and some other depending on your system.

If you want to try it, here is the code you can use:

```
import hashlib

password = "MD5Online"
md5 = hashlib.md5(password.encode())

print("The corresponding hash is : ")
print(md5.hexdigest());
```

By the way, I'm testing this on a Raspberry Pi 4 to make sure it works.
The Raspberry Pi is the perfect device to create a mini server at home (and bring it in travel), to run Python scripts in background. If you want to try decrypting a few passwords, you should definitely consider this affordable solution (more details in my resource page)

Once done, you can use any solution on the market to store the password in a database. For example, you can use MongoDB, MySQL, etc.
I will skip this part as the code highly depends on what you are choosing, and in not the main goal of this tutorial.

Checking the password on login

Once the password is encrypted and stored in database, you can use a simple condition to check that the login attempt you try to validate is correct.

The idea is to compare the input password to the stored password for this user:

```
import hashlib
#The first part depends on the framework you are using
#Let's say you get a password in clear format from the request:
password = "MD5Online"

#The second part depends on the database you are using
#But your password is hashed in the database,
#so you get a string like:
db_password = "d49019c7a78cdaac54250ac56d0eda8a"

#Finally, validate that the two passwords are the same
if (hashlib.md5(password.encode()).hexdigest() == db_password):
   print("Authentication success")
else:
   print("Bad login or password")
   #Probably redirect or display again the login form
```

Do you see the idea?
Just get the two passwords in MD5 format and compare them with a simple condition.
You never need to decrypt the one stored in database, except for hacking, that's what we'll see in the next part.

The best solution to decrypt passwords in Python

If you are still reading these lines, that's because you are here to learn how to really decrypt a list of MD5 passwords and get the plain text as a result.
I have a solution for you.

Hide your IP address and location with a free VPN:
Try it for free now, with advanced security features.
2900+ servers in 65 countries. It's free. Forever.
MD5Online.org is offering an API you can use in Python (or any other language that can handle HTTP requests), to try to decrypt each one of your hashes with our database.

This is a paid service, but it's really affordable, and avoid having a huge server at home that do brute-force all day 🙂
If you want more information, check this page that explains everything.

Once your account is created with a few credits to test (the first package costs €1), **you can get your API key in your account and try this script in Python:**

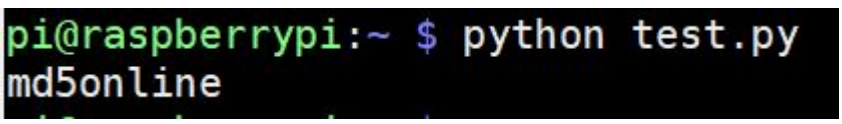
```
#Python Library to make HTTP requests
#(install with 'pip install requests' if needeed)
import requests

#Initialization
url = "https://www.md5online.org/api.php"
key = "YOUR_API_KEY"
md5 = "d3c8e06e57cc1af7ebdba01427e62bc2"

#Request
result = requests.get(url+"?p="+key+"&h="+md5)
print(result.text)
```

If you have any issue with this, you can add the "&d=1" parameter at the end of the URL, to display any error message.
Also, feel free to contact me if you don't know how to fix it.

In any case, it's working well for me:



Conclusion

That's it, you now know how to decrypt MD5 passwords in Python, with the two solutions depending on your situation:

- If the goal is to validate password, you don't need to decrypt them at all (and you know how to do this)
- If your main purpose is try to hack passwords and find the corresponding word, you can use our API at MD5Online.

If this tutorial was useful for you, please share it on your favorite social network 🙂

Whenever you're ready for more security, here are things you should think about:

- **Break free from Gmail**: You should be able to choose what happens to your data. With Proton, only you can read your emails. Get private email.

- **Protect yourself online**: Use a high-speed Swiss VPN that safeguards your privacy. Open-source, no activity logs. Get Proton VPN risk-free.

- **Master Linux commands**: A sure method to learn (and remember) Linux commands. Useful ones only, one at a time, with clear explanations. Download the e-book.