# Introduction to IT Security

## Mandatory assignment 1, part ¾: Cryptography

## Hashes on Linux

We will first introduce how passwords are stored and protected in Linux operating systems.

The following is an example of the `/etc/shadow` file in a Linux system, containing the password hashes of three users called John, Paul and Ringo. The password for both John and Paul is

```
student@ubuntu:~$ sudo tail -n 3 /etc/shadow
john:$6$qIo0foX5$r7kx5FnTYMWANoz8zacMRdHjxiFs9aaKsj2nObFOzS.q86AfOVCxJKH/kqdcDrT
FH9XvSXQ5ZDEmcEzX2NCik/:17447:0:99999:7:::
paul:$6$yoHEm7/a$XUBKbMwYa3V5QPLGL4tsL3yNiGD7Bx5v1grn.sVQfxFp0aLGNPFW51OQYTvtMtE
MNGC.tpBtwu/GPM4SDhp5W.:17447:0:99999:7:::
ringo:$6$yOb0ojJ/$CIHCzyqDq1hhR1fJ3nR9AOiIqvA0XUycbWH1e4QQ/QCt/beFyzFe98AJVoAr/a
LYz2ShRVYwfYY.cKVnnupcP.:17447:0:99999:7:::
```

After each username comes `$6$`, indicating a type 6 password, which means SHA-512 is used. **This hashing algorithm is nowadays not considered to be the best way to protect passwords but it's still allowed in some Linux distributions.** Then there is a long random string of characters that goes up to the next dollar string, which is the *salt*, and then an even longer random string of characters, which is the password hash itself. Check this website to further understand the details of the shadow file.

In this example, both John and Paul have the same password, "P@sw0rd", but their hashes are completely different thanks to the different random salt added to the password before hashing. Moreover, instead of just using one round of the hash function, it uses 5.000 rounds of SHA-512, which takes more CPU time to calculate the hash and thus is meant to slow down attackers who are trying to make dictionaries attacks. It actually has more than 20 steps that involve taking two hashes together and mixing the bits together, but it's a little more complicated than just repeating the same hash algorithm over and over.

Further details of the hashing method can be found in the `/etc/login.defs` file, as shown below.

```
#
ENCRYPT_METHOD SHA512

#
# Only used if ENCRYPT_METHOD is set to SHA256 or SHA512.
#
# Define the number of SHA rounds.
# With a lot of rounds, it is more difficult to brute forcing the password.
# But note also that it more CPU resources will be needed to authenticate
# users.
#
# If not specified, the libc will choose the default number of rounds (5000).
# The values must be inside the 1000-999999999 range.
# If only one of the MIN or MAX values is set, then this value will be used.
# If MIN > MAX, the highest value will be used.
#
# SHA_CRYPT_MIN_ROUNDS 5000
# SHA_CRYPT_MAX_ROUNDS 5000

################ OBSOLETED BY PAM ##############
#                                              #
# These options are now handled by PAM. Please  #
# edit the appropriate file in /etc/pam.d/ to   #
--More--(92%)
```

In this exercise, we'll be using the Python library passlib. Once installed, you need to import the `sha512_crypt` module. Here's how you can use it:

```
>>> import hashlib
>>> from passlib.hash import sha512_crypt
>>> sha512_crypt.using(salt="qIo0foX5", rounds=5000).hash("P@sw0rd")
'$6$qIo0foX5$r7kx5FnTYMWANoz8zacMRdHjxiFs9aaKsj2nObFOzS.q86AfOVCxJKH/kqdcDrTFH9XvSXQ5ZDEmc
zX2NCik/'
>>>
```

As you can see, we get the correct results (starting with `r7k`), that is the hash of John's password.

If we were to do a dictionary attack, we would have a series of password guesses as shown below:

```
>>> sha512_crypt.using(salt="qIo0foX5", rounds=5000).hash("P@sw0ra")
'$6$qIo0foX5$DX/.gSq2C8NdzBK9Yn2lWTLMw3nQw1pRebYEBaasuyEioRFImk9tF1cHIlC9Ecmdnk6QZFuYnkUXWk
.6Jssbo0'
>>> sha512_crypt.using(salt="qIo0foX5", rounds=5000).hash("P@sw0rb")
'$6$qIo0foX5$iYl6BYzTCgWTdRl5ErYB/EoL4ImlwwPBMqDrLDSVomCvLG83Id8oAAkrFixSzxoKtmu79qBLa6JDK6
irIMwLj0'
>>> sha512_crypt.using(salt="qIo0foX5", rounds=5000).hash("P@sw0rc")
'$6$qIo0foX5$FJnTG1I0aaR2Z7MadTzSG90WaL57vC8Jta./DKLw9f6vhHdi9BXThneeocx0e5hYKTJJIrT17uOzkB
WNwGhdQ/'
>>> sha512_crypt.using(salt="qIo0foX5", rounds=5000).hash("P@sw0rd")
'$6$qIo0foX5$r7kx5FnTYMWANoz8zacMRdHjxiFs9aaKsj2nObFOzS.q86AfOVCxJKH/kqdcDrTFH9XvSXQ5ZDEmcE
zX2NCik/'
>>>
```

So it's just a question of keep trying different potential passwords until we get the one that matches the hash found in the shadow file. Since SHA-512 has been used instead of a password-base KDF, it's in general feasible to execute the attack successfully in a reasonable amount of time.

Now that we've reviewed Linux hashes, we're ready to present the exercise.

**The shadow file of a Linux operating system contains the following line:**

`"$6$penguins$eP.EvNiF2A.MmRVWNgGj5WSXKK8DAf7oeK8/kkbollee.F0T4KAy.QEgNAX.6wLQY1XHmSID/5VkeFiEaSA2b0"`

**You're asked to crack the given hash, knowing that the password is in a 3-digit format. You're encouraged to solve this exercise using Python. Once you're done, and if you want, you can use a password-cracking tool like John the Ripper to confirm the result.**