# Частина 1:

Генератор створює різні типи паролів з певною імовірністю:

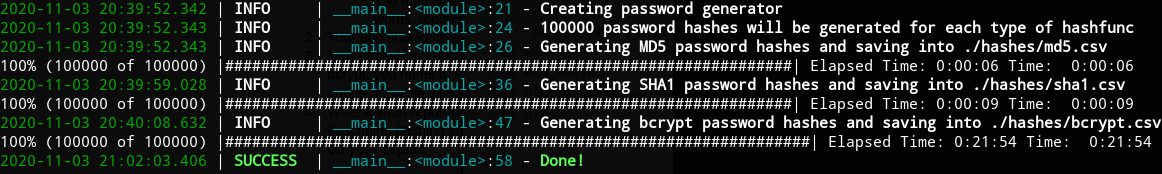
* 75% - береться пароль з топ 100000 найпопулярніших паролів
* 10% - беруться з топ 110 найпопулярніших паролів
* 10% - схожі на справжні, генеруються за допомогою найчастіше використовуваних слів, цифр та спецсимволів
* 5% - генеруються випадковим чином

Випадкові паролі генеруються довжиною від 6 до 16 символів, кожен з яких рандомно обирається серед латинських букв, цифр та символів пунктуації

Схожі на справжні паролі генеруються наступним чином:

1. Обирається випадкова довжина від 6 до 16 символів
2. Поки довжина недостатня додаємо
   1. З імовірністю 65% слово з найвживаніших англійських слів(З імовірністю 35% перша буква слова змінюється на букву в верхньому регістрі)
   2. З імовірністю 25% випадкова цифра
   3. З імовірністю 10% додається один із спецсимволів (\*\_.!+-)
3. Якщо довжина перевищує обрану, обрізаємо зайві символи

Для кожного з типів хешування (MD5, SHA1, bcrypt) ми створюємо 100000 паролів за допомогою генератора, хешуємо їх відповідним чином та зберігаємо в файлі csv. Для SHA1 поряд в файлі зберігаємо сіль.



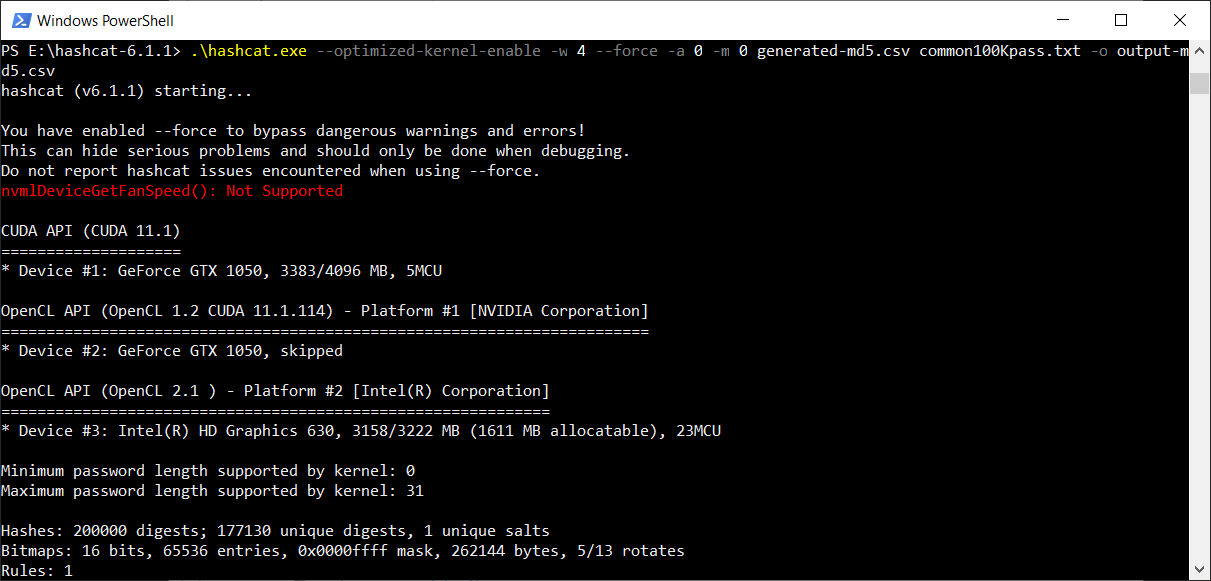
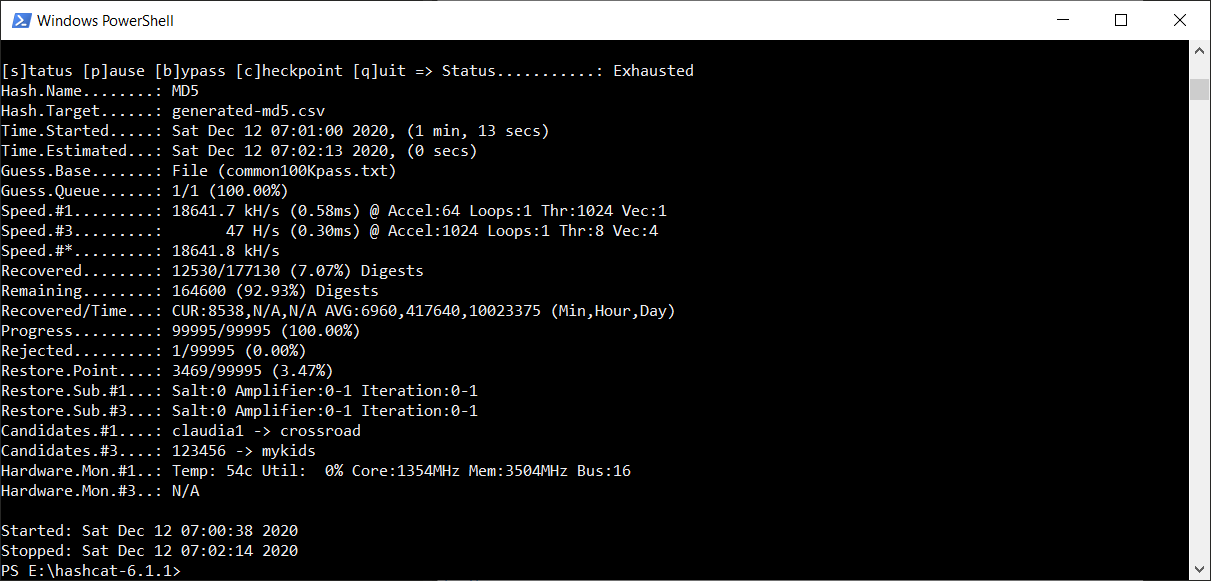
# Частина 2

Згенеровані паролі були взяті з <https://github.com/vladlytvynenko/crypto-labs/tree/master/lab4>

## MD5

Спробуємо розшифрувати паролі захешовані за допомогою MD5, для будемо використовувати список з топ 100 тисяч найпоширеніших паролів. Для цього будемо використовувати "straight" mode (Dictionart attack). Він просто перебирає всі паролі що вказані у заданому файлі і намагається знайти такий же хеш у заданому файлі з хешами невідомих паролів.

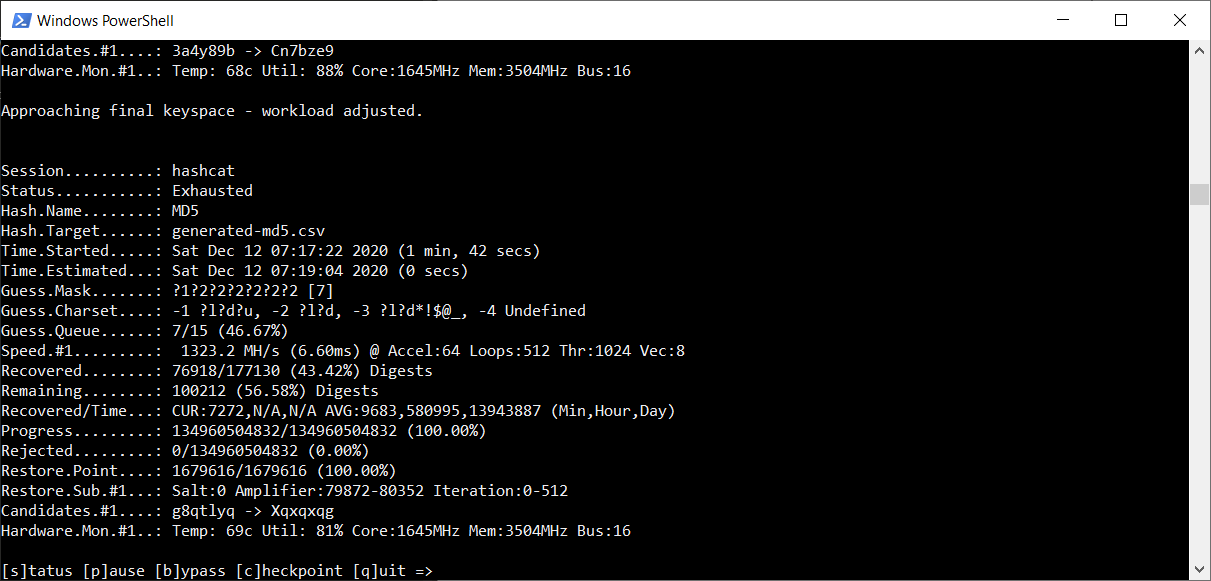
***hashcat --optimized-kernel-enable -w 4 --force -a 0 -m 0 generated-md5.csv common100Kpass.txt -o output-md5.csv***

******

Було відновлено 12530 з 177130 паролів, тобто біля 7 відсотків паролів.

Тепер спробуємо ще раз тільки цього разу методом брутфорс із стандартними налаштуваннями.

***hashcat --optimized-kernel-enable -w 3 -a 3 -m 0 generated-md5.csv -o output-md5-brute.csv***



Цього разу вже було відновлено 43 відсотки (76918/177130) паролів всього за 2 хвилини.

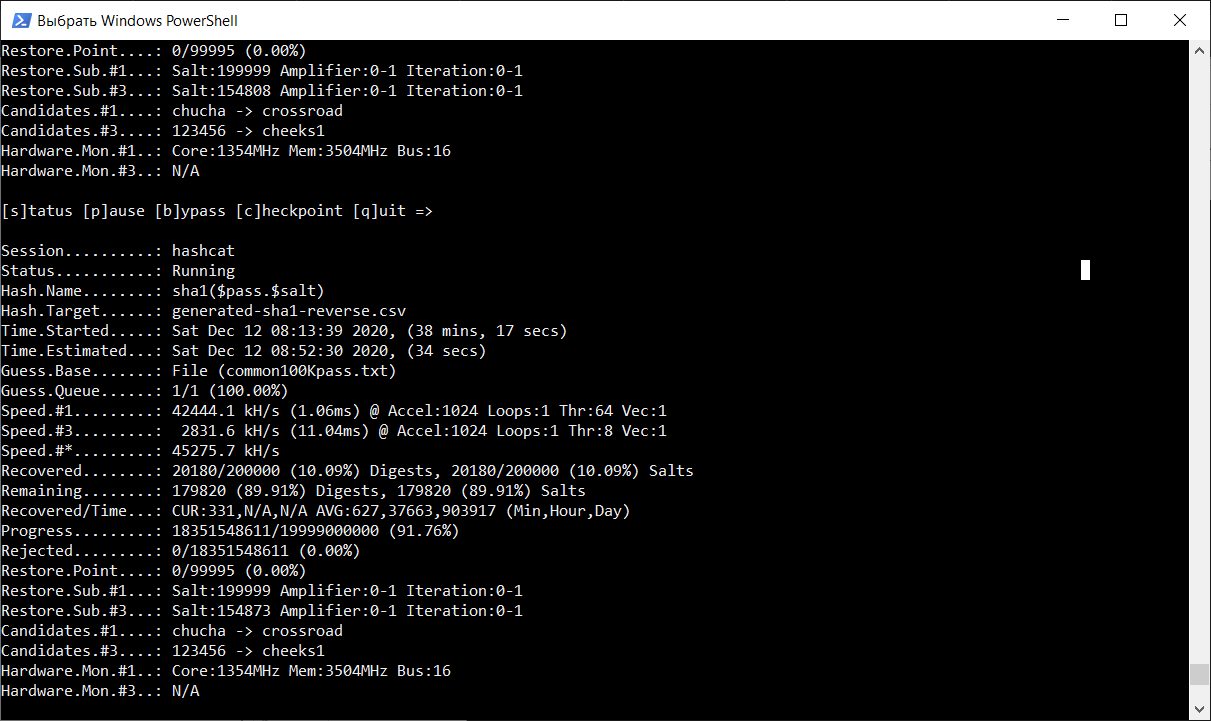
## SHA1 + salt

Перейдемо до паролів, що захешовані за допомогою SHA1 + salt

Для цього використаємо скрипт adapter.py, який просто міняє у файлі місцями сіль та хеш.

Тепер пробуємо відновити паролі за допомогою найпопулярніших паролів.

***hashcat -w 3 -d 1 -a 0 -m 110 generated-sha1-reverse.csv common100Kpass.txt -o output-sha1.csv***



Після майже 40 хв. роботи було розшифровано 10 відсотків паролів. Такий достатньо великий час для цього випадку через те що по середині обчислень дискретна відеокарта перестала відповідати і решту обчислень проводилися лише на інтегрованій.

Запускаємо ще раз і маємо результат: після 6 хвилин роботи було відновлено 11% паролів. Методом dictionary.

Пробуємо брутфорс

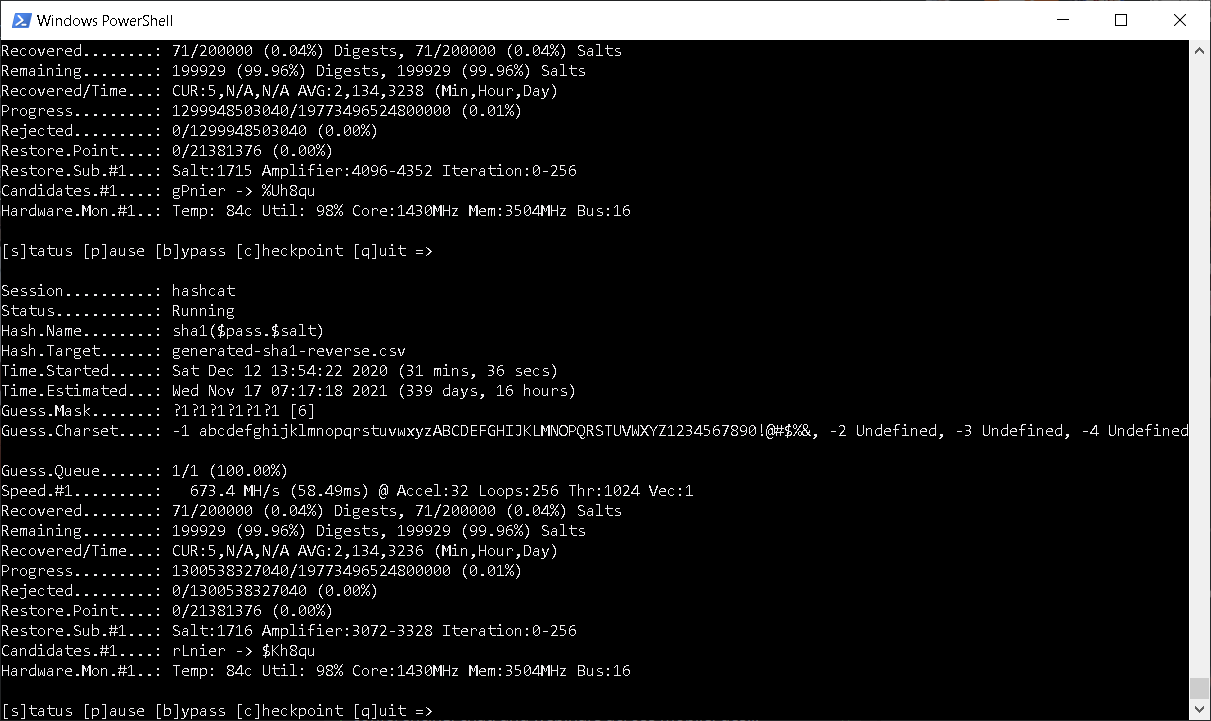
Використовуючи брутфорс скористаємося знанням про те з яких символів складаєтсья пароль та в якому проміжку буде його довжина.

Набір символів:

**abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%&**

Спробуєммо знайти паролі довжиною в 5 символів.

***hashcat.exe -w 3 -d 1 -a 3 -m 110 -o output-sha1-brute.csv -1 "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%&" generated-sha1-reverse.csv ?1?1?1?1?1?1***

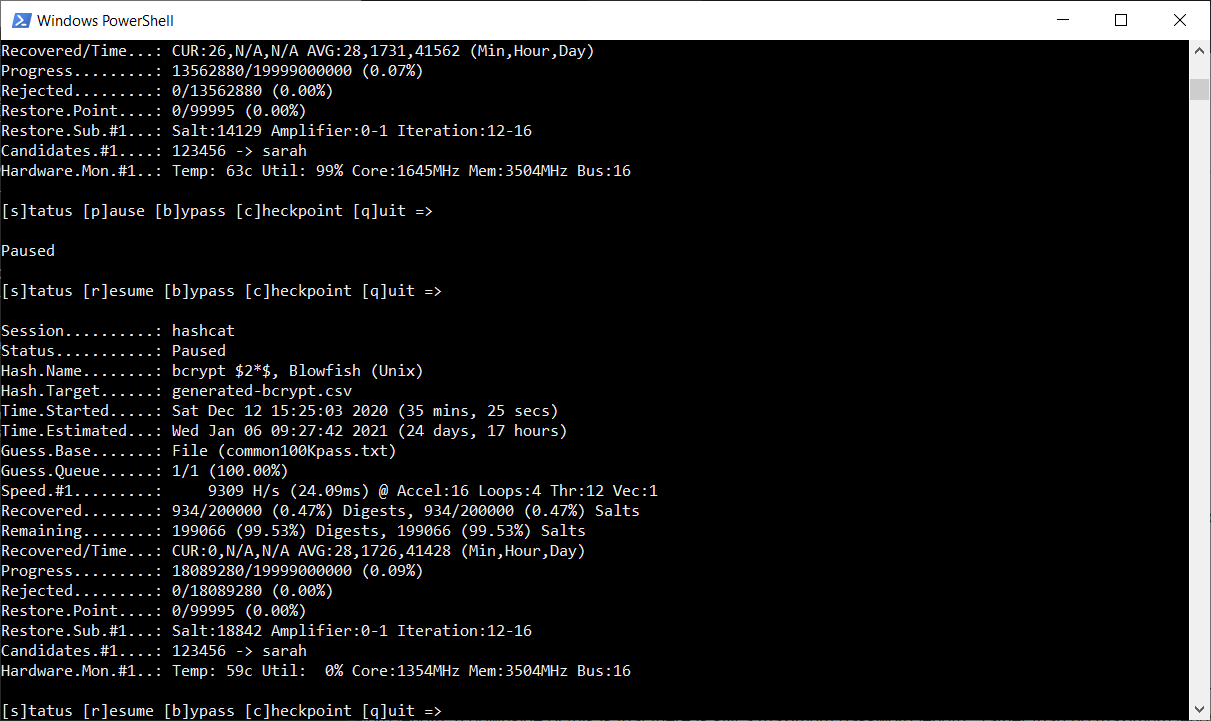


Можемо спостерігати що після 30 хвилин було розшифровано всього 71 пароль (і це при швидкості 670 MH/s). Для повного перебору знадобиться біля 350 днів.

## bcrypt

Пробуємо відновити паролі методом dictionary. З кількістю раундів для генерації солі, що дорівнює 4.

***hashcat.exe -w 3 -d 1 -a 0 -m 3200 generated-bcrypt.csv common100Kpass.txt -o output-bcrypt.csv***



За 35 хвилин відновилося всього 934 паролі (0.47%)

## Підсумок

Як бачимо MD5 проявив себе найгірше. За дуже короткий час dictionary відновлює достатньо велику кількість паролів(7%), а перебір відновлює взагалі 43% всього за 2 хвилини.

Sha1 + salt дає значно кращі результати, проте теж дуже вразливий до методу dictionary. Але майже не піддається brute-force атакам.

Bcrypt показує найкращий результат, він достатньо стійкий як до dictionary так і до brute-force.

Найлегше відновлювати паролі що вже є в словниках з найбільш використовуваними, важко зламати ті, що мають велику довжину.

Тому на паролі краще вводити наступні обмеження:

* Велика кількість символів(мінімум 8)
* Якнайбільше різних типів символів(літери як верхнього так і нижнього регістру, цифри та різні спецсимволи)
* Не допускати паролі які входять до найбільш популярних

Для хешування не можна використовувати методи типу MD5. Краще використовувати стійки алгоритми типу bcrypt, scrypt та argon2.