Брутфорс хэшей в Active Directory / Хабр

habr.com/ru/companies/deiteriylab/articles/584160

kukuxumushi

kukuxumushi 19 окт 2021 в 19:34

Брутфорс хэшей в Active Directory

19 мин

34K
Туториал

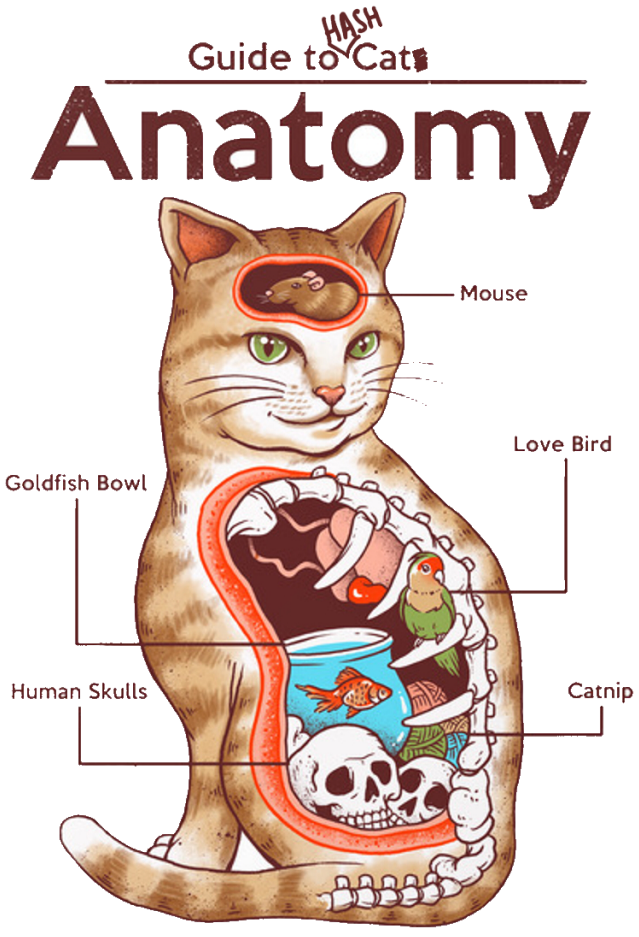


Image retrieved from https://www.redbubble.com/people/vincenttrinidad/shop

Слабые пароли пользователей — очень распространенная проблема, которая может позволить злоумышленнику повысить свои привилегии и закрепиться в сети компании.

Чтобы этого не допустить, необходимо регулярно анализировать стойкость паролей пользователей.

У системных администраторов нет возможности увидеть пароли пользователей в открытом виде, но они могут выгрузить хэши паролей и провести их перебор.

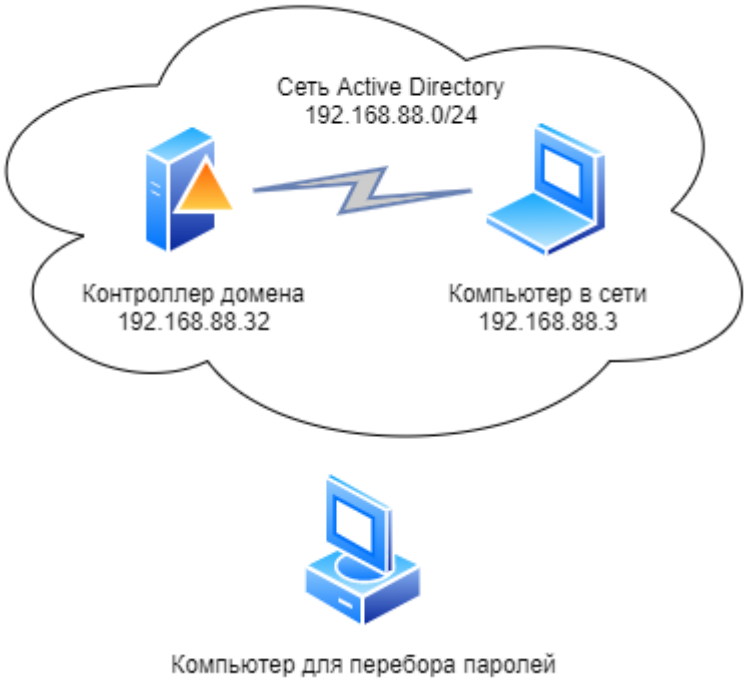
В этой статье мы расскажем о том, как можно легко получить хэши паролей пользователей и провести эффективный перебор паролей, используя различные методы.

Данный материал также будет полезен пентестерам, которым для развития атаки требуется восстановить пароли пользователей из их хэш-значений.

▶ TL;DR

Конфигурация тестового стенда

Для демонстрации получения хэшей была создана Active Directory инфраструктура, состоящая из одного контроллера домена и одного компьютера в сети.



Конфигурация компьютера для перебора паролей:

- Intel Core i7-7700.
- 32GB RAM.
- Nvidia GeForce GTX 1060 6GB.
- hashcat 6.1.
- Windows 10 build 19041.572.
- GeForce Game Ready Driver 460.89.

- CUDA 11.2.0.

Подготовка

Как получить хэши для перебора?

▶ Что такое NTLM-хэш?

Для получения хэшей всех пользователей домена можно воспользоваться утилитой secretsdump.py, входящей в состав пакета impacket. Для использования утилиты необходимо иметь установленный интерпретатор языка программирования Python версии 2.7 или выше. Также существуют сборки утилиты secretsdump, специально созданные для работы под OS Windows, и не требующие наличия сторонних интерпретаторов.

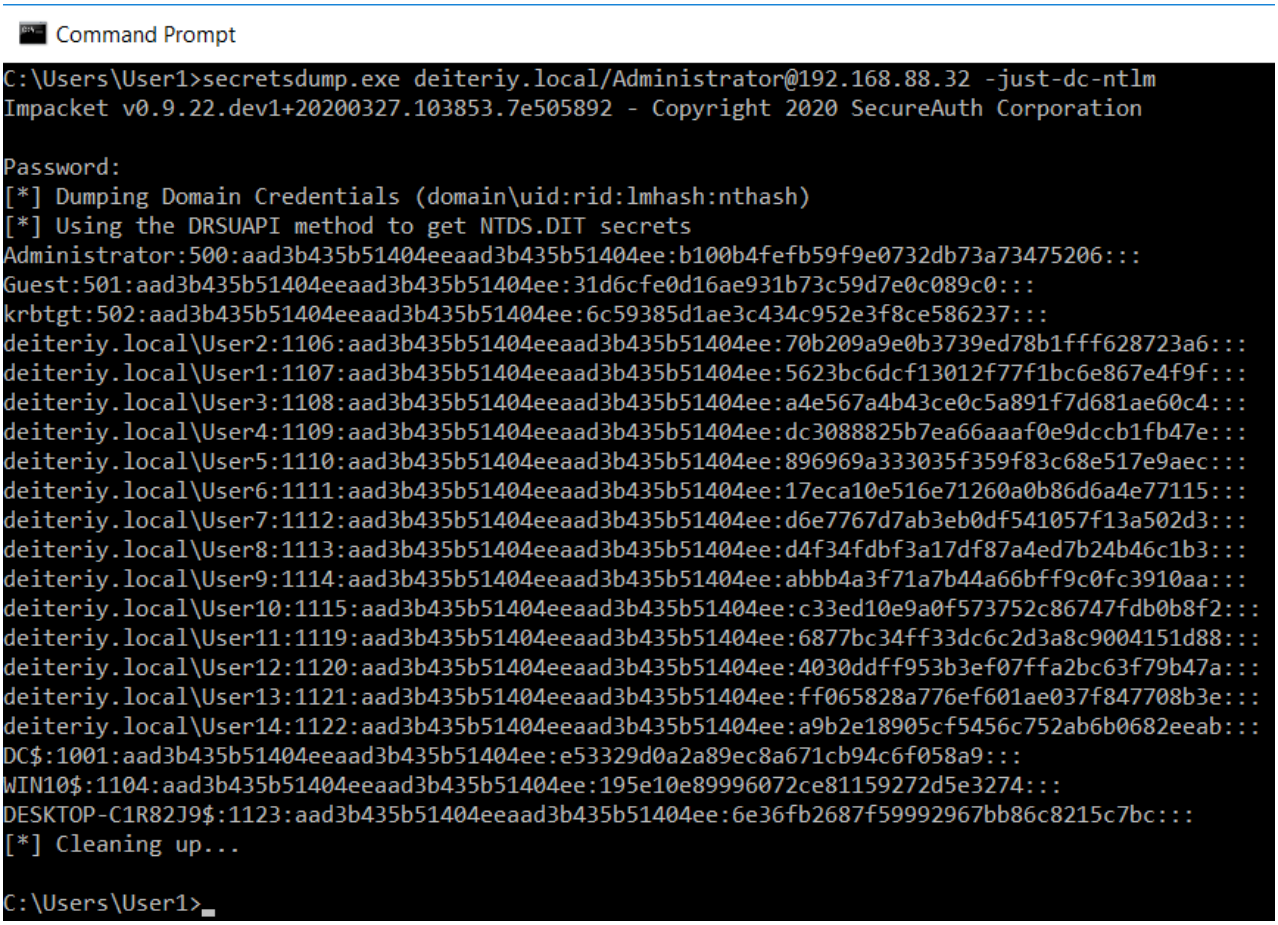
Данная утилита может быть запущена на любом устройстве, имеющем сетевой доступ к контроллеру домена.

Для получения NTLM-хэшей с контроллера домена, можно воспользоваться следующей командой. Данная команда может быть выполнена на любом устройстве, имеющем сетевой доступ к контроллеру домена.

```
secretsdump.exe deiteriy.local/Administrator@192.168.88.32 -just-dc-ntlm
```

где:

- `deiteriy.local` — название домена.
- `Administrator` — имя учетной записи с правами доменного администратора.
- `192.168.88.32` — IP-адрес или доменное имя контроллера домена.
- `-just-dc-ntlm` — аргумент, позволяющий получить только NTLM-хэши в формате domain\uid:rid:lmhash:nthash.



Пример успешного получения NTLM-хэшей с тестового контроллера домена

Чем перебирать?

Offline:

- hashcat — Самая популярная и производительная утилита, поддерживающая множество различных алгоритмов хэширования. Данная статья посвящена этой программе.
- John the Ripper — Старейший набор программ для перебора хэшей и преобразования различных файлов в их хэш-значения для последующего перебора. Кросплатформенный, поддерживает множество алгоритмов, однако имеет меньшую производительность, чем hashcat.
- RainbowCrack — Данное приложение использует другой подход к перебору хешей: не вычисление хэшей от паролей и их последующее сравнение, а поиск хэшей по заранее сгенерированным таблицам, содержащим пары хэш — значение. Отличается очень высокой скоростью подбора (быстрее в ~5000 раз), по сравнению с остальными утилитами, однако требует хранения заранее сгенерированных таблиц для каждого алгоритма хэширования, которые занимают **десятки терабайт** дискового пространства.

Online:

- crackstation — Сервис, позволяющий производить поиск хэшей в словаре, содержащем 1,6 миллиарда паролей.
- onlinehashcrack — Сервис, предоставляющий возможность полного перебора и перебора паролей по словарю в облаке. При успешном восстановлении исходного значения, будет предложено оплатить услуги для получения этого значения. Для проверки сложности паролей, может быть достаточно информации о том, что исходное значение хэша можно восстановить за приемлемое время.
- Поисковые системы (Google, Yandex, …) — Любой хэш можно “загуглить”. Если в поисковой выдаче есть проиндексированная страница с данным хешем, то велик шанс, что на ней находится соответствие хэш — исходное значение.

Установка hashcat

1. Скачать в GitHub репозитории последнюю сборку приложения в архиве. Архив содержит программу под операционные системы Windows и Linux.
2. Скачать и установить последнюю версию драйверов GPU (Nvidia, AMD).
3. Разархивировать файл из первого пункта и запустить hashcat.exe из командой строки. Для проверки корректности установки можно использовать следующую команду:

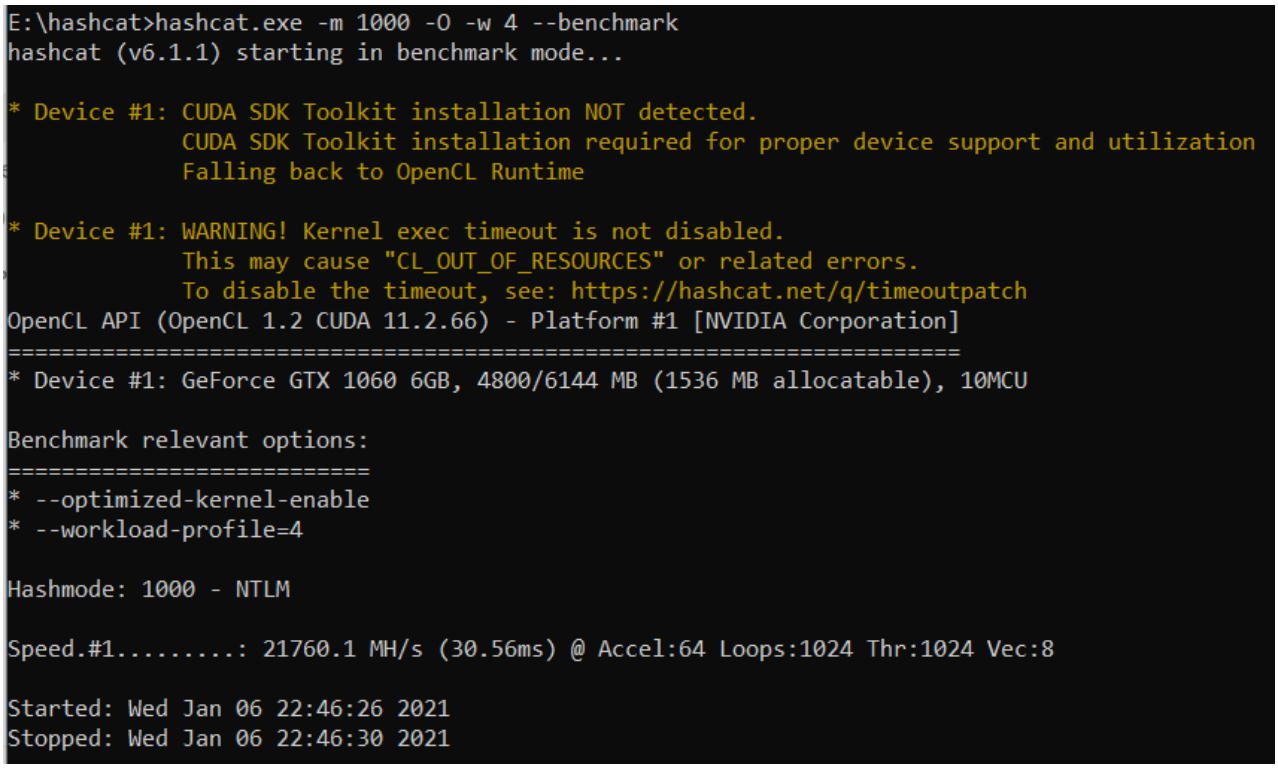
   ```
   hashcat.exe -w 4 -O -m 1000 --benchmark
   ```

   где:

   - `-w 4` — Выбор режима рабочей нагрузки из заданного набора, где 1 — самая минимальная нагрузка, а 4 — максимальная. Если вы заметили существенное ухудшение производительности, которое мешает работе с устройством, рекомендуется использовать меньшее значение параметра -w.
   - `-O` — Включить оптимизированные ядра.
   - `-m 1000` — Выбор алгоритма хаширования, в данном случае 1000 — NTLM.
   - `-benchmark` — Запустить тестирование производительности для выбранного алгоритма.

   Более подробно об аргументах можно прочитать на странице помощи утилиты:

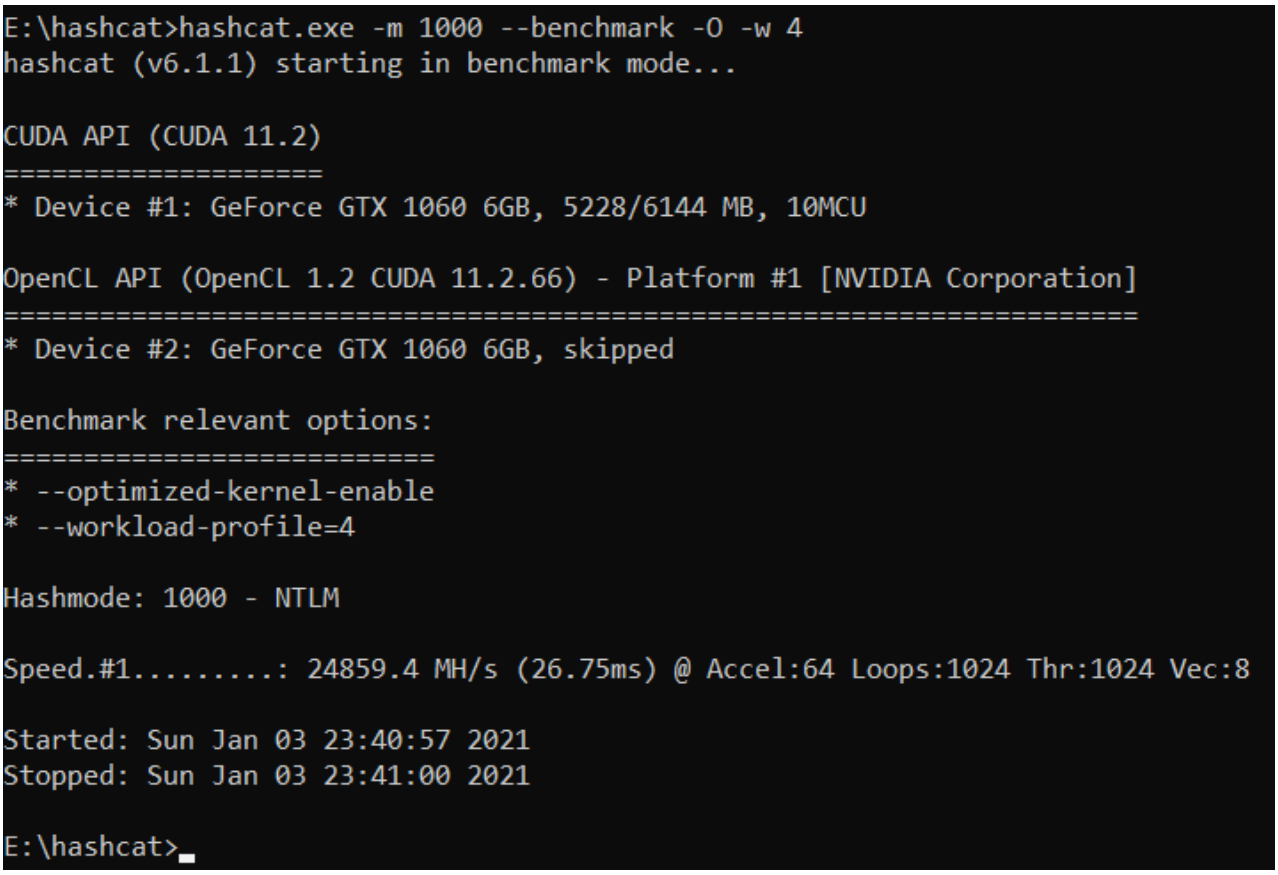
   ```
   hashcat.exe --help
   или
   hashcat.exe -h
   ```



Пример успешного запуска утилиты hashcat в режиме benchmark

Если используется GPU Nvidia, то можно увидеть предупреждение, как на рисунке выше. По умолчанию hashcat использует открытый фреймворк OpenCL для работы с GPU, однако Nvidia разработала свой собственный — CUDA. Рекомендуется установить его для повышения производительности.

Запустив еще раз команду выше, видим, что после установки CUDA, производительность выросла на 15%.



Пример успешного запуска утилиты hashcat в режиме benchmark с установленным CUDA драйвером

Перебор паролей

90% успеха в переборе паролей — это правильно выбранный метод подбора. В данной статье рассмотрено несколько таких методов, а в конце приведено сравнение таких методов по скорости и эффективности.

Перебор по публично доступным словарям

Несколько примеров словарей, которые можно найти в открытом доступе:

- SecLists — Репозиторий содержит большое количество файлов с паролями, в том числе рейтинги самых популярных паролей, а также словарей, полученных в результате утечек из различных сервисов.
- Crackstation — Сервис, позволяющий производить поиск хэшей в словаре, предоставляет один из самых больших (1,6 миллиарда строк) словарей. По заявлению создателей, словарь содержит все пароли, которые когда-либо были опубликованы в Интернете.
- Probable-Wordlists — Репозиторий содержит словари паролей как из утечек, так и из различных публично доступных словарей. Проект содержит аналитику популярности использования паролей, а также маски и правила для утилиты hashcat.

- weakpass.com — На сайте собрано большое количество словарей агрегированных из различных источников и их комбинаций. Есть словари размером как меньше мегабайта, так и больше четырехсот гигабайт.

Словарь rockyou.txt

rockyou.txt — самый популярный словарь для перебора, состоящий из самых популярных паролей, собранных из публично опубликованных баз данных.

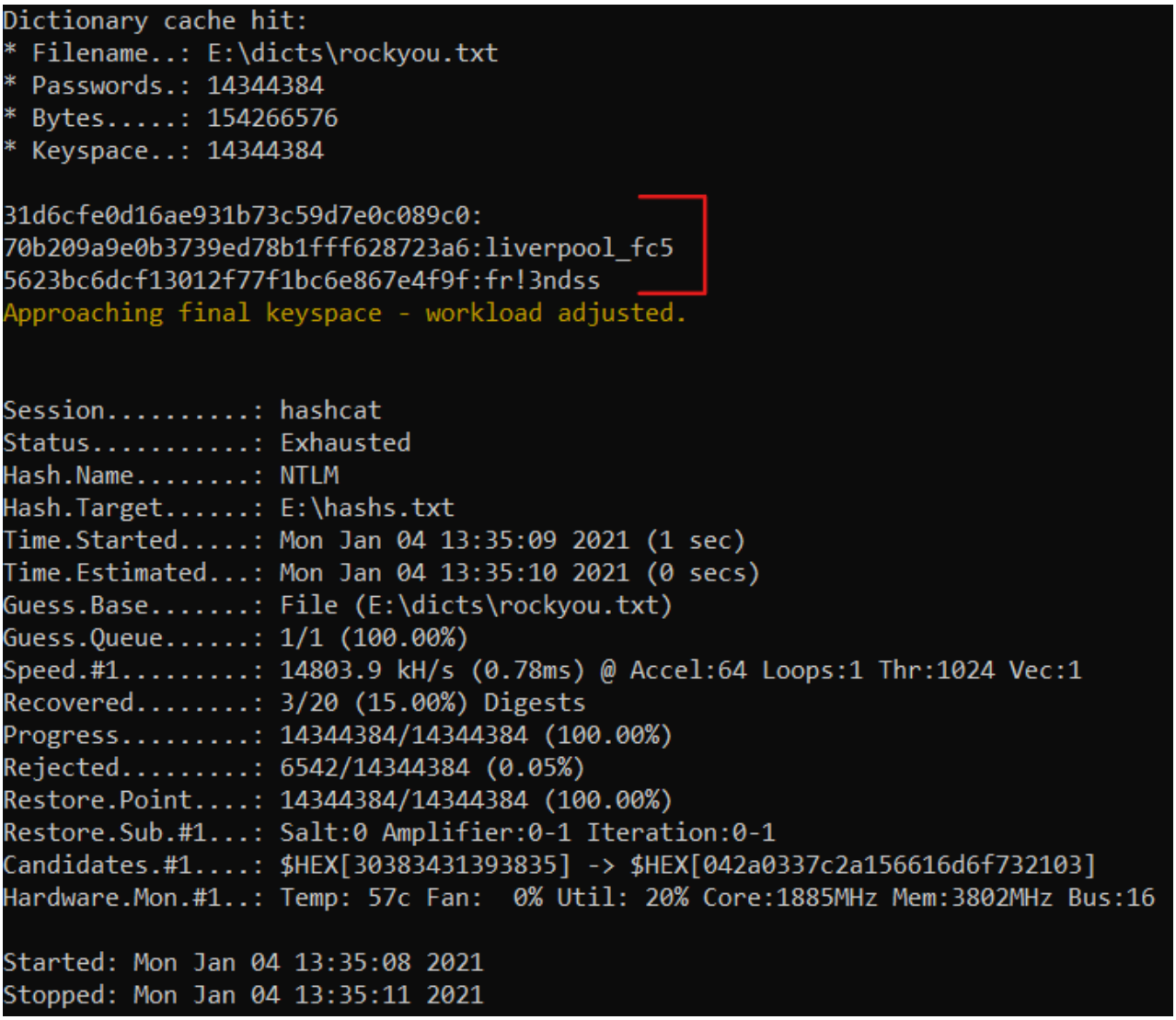
Для перебора по словарю rockyou.txt, его необходимо загрузить на устройство, а затем выполнить следующую команду в командной строке:

```
hashcat.exe -m 1000 -a 0 -O -w 4 E:\hashs.txt E:\dicts\rockyou.txt
```

где:

- `-m 1000` — Выбор алгоритма хэширования, в данном случае 1000 — NTLM.
- `-a 0` — Выбор режима работы (0 — перебор по словарю, 3 — перебор по маске, 1,6,7 — Гибридные режимы работы).
- `-O` — Включить оптимизированные ядра.
- `-w 4` — Выбор режима рабочей нагрузки из заданного набора 1-4, где 1 — самая минимальная нагрузка, а 4 — максимальная.
- `E:\hashs.txt` — Путь к файлу с хэшами.
- `E:\dicts\rockyou.txt` — Путь к словарю.

На рисунке ниже, видим пример вывода программы. Красным выделены пары хэш — значение, которые были успешно восстановлены.



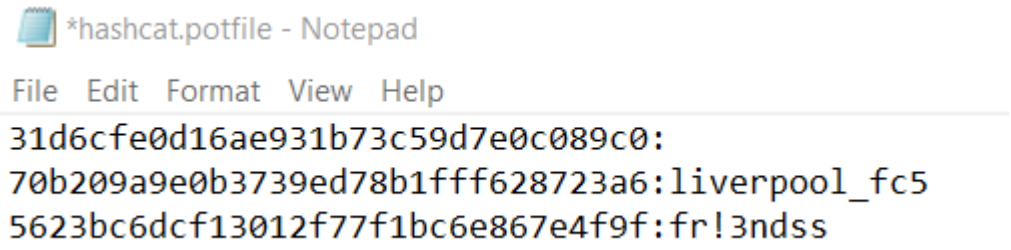
Пример успешно подобранных паролей

На тестовом стенде программе потребовалось 3 секунды на перебор 20-и хэшей по словарю, содержащему 1,4 миллиона паролей.

После успешного подбора, пары хэш — значение будут помещены в pot-файл, расположенный в папке с программой — E:\hashcat\hashcat.potfile. При каждом запуске программы, в первую очередь будет проверяться наличие хэшей в pot-файле, чтобы не перебирать известные хэши заново. Если такой хэш был найден, то он не будет выведен на экран. Вместо этого будет выведено информационное сообщение о том, что такой хэш был найден в pot-файле.

```
INFO: Removed 3 hashes found in potfile.
```

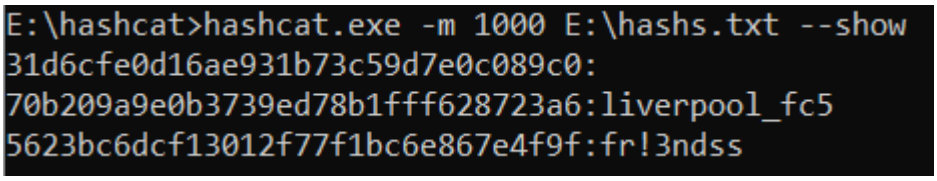
Содержимое pot-файла:



Пример содержимого pot-файла

Для просмотра таких хэшей, можно воспользоваться командой:

```
hashcat.exe -m 1000 E:\hashs.txt --show
```



Поиск хэшей в pot-файле

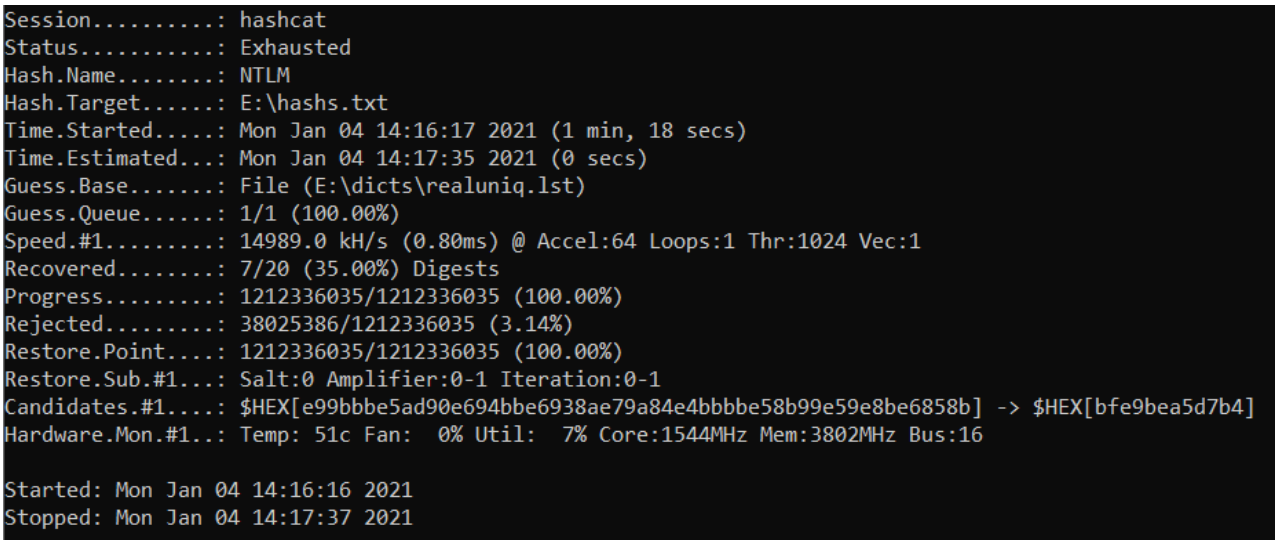
Словарь realunique от Crackstation

realuniq.lst — По заявлению создателей — это самый большой файл с паролями на просторах интернета — содержит 1,6 миллиарда паролей.

Для перебора по словарю realunique, его необходимо загрузить на устройство, а затем выполнить следующую команду в командной строке:

```
hashcat.exe -m 1000 -a 0 -O -w 4 E:\hashs.txt E:\dicts\realuniq.lst
```

Результат работы на рисунке ниже:



Перебор по словарю

На тестовом стенде программе потребовалось 39 секунд на перебор 20-и хэшей по словарю, содержащему 1,2 миллиарда паролей.

Перебор по самодельному словарю

У каждой организации свои парольные политики и своя культура, связанная с особенностями работы (Bank2020!, Neft2020@, C@rd2019), местоположением (Russia2020!, Astana2010#, Moscow123$), названиями компании (BankRot123, G@zM0sPr0m, Vodorod!@#), филиалов (H3adOff1ce1, $KolymaDep$, Branch*) и отделов (HR123!@#, ITdep!!!, Accountants200). Также словари не учитывают специфику языка пользователей (Mamir2020 (май на казахском языке), Vfq2020! (май на русском языке, набранный другой раскладкой), Maijs2018 (май на латышском языке)). Поэтому рекомендуется создавать свои собственные словари для перебора паролей и использовать их вместе с другими методами перебора.

Также, публичные словари редко учитывают специфику сетей Windows и ее парольные политики:

Парольная политика "Passwords must meet complexity requirements", регламентирует, что пароль должен содержать символы как минимум из трех следующих категорий:

1. Заглавные буквы европейских языков (от A до Z, с диакритическими знаками, греческие и кириллические символы).
2. Строчные буквы европейских языков (от a до z, ß, с диакритическими знаками, греческие и кириллические символы).
3. Цифры (от 0 до 9).
4. Не алфавитно-цифровые символы (специальные символы): (`~! @ # $% ^ & * _- + = \| \ () {} [] :;" '<>,.? /`) Символы валюты, такие как евро или Британский фунт не считаются специальными символами для этого параметра политики.
5. Любой символ Юникода, относящийся к категории алфавитных символов, но не прописных или строчных. В эту группу входят символы Unicode из азиатских языков.

Также, стоит учитывать, что в Windows по умолчанию стоит минимальная длина пароля 8 символов, а частота смены пароля обычно установлена 90 дней.

Словарь — календарные единицы

При частой смене паролей пользователям сложно запомнить сложный, стойкий пароль, который при этом будет действовать всего 3 месяца, и будет существенно отличаться от предыдущего. Поэтому, зачастую пользователи используют пароли, состоящие из следующих частей:

- Месяц
- Время года

- Год
- Спецсимволы

Например:

- May2020!
- Leto2019
- Aprel2021@
- 19October.
- Pbvf2020!! (где “Pbvf” — это слово “Зима” набранное другой раскладкой)

Для генерации такого словаря можно воспользоваться небольшим скриптом на Python:

```
importdatetime
fromitertoolsimportpermutations, product#библиотека для комбинирования элементов



#определяем компоненты, из которых будет состоять пароль
specials= ['!','@','#','$','.','*','^','?'] #спецсимволы



#месяца и времена года
words= ["April", "Atdhfkm", "August", "Autumn", "Besna", "Becna", "B.km", "B.ym",
"Ctynz,hm", "December", "Dtcyf", "Fduecn", "February", "Fghtkm", "January",
"Jctym", "Jrnz,hm", "July", "June", "Ktnj", "Leto", "Ltrf,hm", "March", "May",
"November", "October", "Ocen", "Osen""Pbvf", "September", "Spring", "Summer",
"Vesna", "Vecna", "Vfq", "Vfhn", "Winter", "Yjz,hm", "Zima", "Zydfhm", "3ima"]
# года с 2010 по текущий + 2 в форматах YYYY и YY
now=datetime.datetime.now()
years=list(range(2010,now.year+2))+list(range(10,int(str(now.year)[2:])+2))



#открываем файл для сохранения паролей
f_o=open("calendar_passwords.txt", "w")



#генерируем список специальных символов от 0 до 3 элементов
special_array=[]
foriinrange(0,4):
forsubsetinproduct(specials,repeat=i):
special_array.append(''.join(subset))



#собираем компоненты в пароль
forwordinwords:
foryearinyears:
forspecialinspecial_array:
forpasswordinpermutations([word, str(year), special], 3):
f_o.write(''.join(password)+"\r\n")

                                                    #закрываем файл
f_o.close()
```

В результате получаем словарь на 3,2 миллиона паролей.

```
hashcat.exe -m 1000 -a 0 -O -w 4 E:\hashs.txt E:\dicts\calendar_passwords.txt
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: E:\hashs.txt
Time.Started.....: Mon Jan 04 17:53:24 2021 (0 secs)
Time.Estimated...: Mon Jan 04 17:53:24 2021 (0 secs)
Guess.Base.......: File (E:\dicts\calendar_passwords.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 18582.2 kH/s (0.86ms) @ Accel:64 Loops:1 Thr:1024 Vec:1
Recovered........: 2/20 (10.00%) Digests
Progress.........: 3201120/3201120 (100.00%)
Rejected.........: 0/3201120 (0.00%)
Restore.Point....: 3201120/3201120 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: ^.?Vecna2012 -> ???213ima
Hardware.Mon.#1..: Temp: 57c Fan:  0% Util: 25% Core:1708MHz Mem:3802MHz Bus:16

Started: Mon Jan 04 17:53:23 2021
Stopped: Mon Jan 04 17:53:25 2021
```

Перебор по словарю

На тестовом стенде программе потребовалось 2 секунды на перебор 20-и хэшей по словарю, содержащему 3,2 миллиона паролей.

Пароль-“змейка”

“Змейка” это строка, набираемая на клавиатуре, позиции символов в которой, подвержены неким закономерностям, например, набираются друг за другом (qwerty, asdfgh, zxcvb), находятся друг под другом (qaz, wsx, edc), их можно набирать “не отрывая руки” от клавиатуры (qwedcxza, zxcdsaqwe, 4rfvbgt5) или комбинации этих закономерностей (1qaz2wsx3edc).

Большое число таких паролей содержится в публичных словарях, однако не все удовлетворяют парольным политикам Windows.

Для генерации таких паролей можно воспользоваться утилитой kwprocessor, либо набрать “змейку” самостоятельно, учитывая особенности парольных политик (!QAZ2wsx#EDC, 1q2w3e4r5t^Y, !QAZ1qaz!QAZ).

Перебор по словарю с правилами мутации

Правила мутации (Rule-based Attack) — преобразование слов из словаря, по некоторым, заранее заданным, правилам.

Для описания правил в hashcat существует специальный язык программирования. Написание собственных правил — это нетривиальная задача, поэтому hashcat уже содержит несколько десятков наборов правил, найти которые можно в директории “rules”. Также существует множество наборов правил в открытом доступе, например в репозиториях на GitHub.

dive.rule

dive.rule — самый большой набор правил, из тех, что идет в составе hashcat.

Для запуска hascat со словарем rockyou.txt и этим набором правил можно воспользоваться следующей командой:

```
hashcat.exe -m 1000 -a 0 -O -w 4 E:\hashs.txt E:\dicts\rockyou.txt -r
rules\dive.rule
```

где:

- `-r` — Использовать правила.
- `rules/dive.rule` — Путь к правилам.

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: E:\hashs.txt
Time.Started.....: Mon Jan 04 18:17:36 2021 (6 mins, 53 secs)
Time.Estimated...: Mon Jan 04 18:24:29 2021 (0 secs)
Guess.Base.......: File (E:\dicts\rockyou.txt)
Guess.Mod........: Rules (rules/dive.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3350.8 MH/s (44.61ms) @ Accel:64 Loops:256 Thr:1024 Vec:1
Recovered........: 13/20 (65.00%) Digests
Progress.........: 1421327633024/1421327633024 (100.00%)
Rejected.........: 648220612/1421327633024 (0.05%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:99072-99086 Iteration:0-256
Candidates.#1....: $HEX[303834313938352e67] -> $HEX[042a380337c2a156616d6f732103042a380337c2a156616d6f7321]
Hardware.Mon.#1..: Temp: 61c Fan: 68% Util:100% Core:1873MHz Mem:3802MHz Bus:16

Started: Mon Jan 04 18:17:35 2021
Stopped: Mon Jan 04 18:24:30 2021
```

Перебор по словарю с правилами мутации

На тестовом стенде программе потребовалось 415 секунд на перебор 20-и хэшей по словарю, из которого было создано 1,4 триллиона паролей. Также видим, что словарь, который содержал всего 3 пароля из 20, с помощью правил позволил восстановить 13 из 20 значений.

Получены с использованием правил	Получены без использования правил
(пустой пароль)	(пустой пароль)
liverpool_fc5	liverpool_fc5
fr!3ndss	fr!3ndss
!QAZ1qaz!QAZ	
118@4	

Albert93!
May2020!
!QAZ2wsx#EDC
(!)(!)
Ktnj2020
987654cc–
1q2w3e4r5t^Y
2020jana@@

OneRuleToRuleThemAll.rule

OneRuleToRuleThemAll.rule — набор правил, который является комбинацией самых популярных правил из других наборов.

По заверениям создателей, при двукратно меньшем размере позволяет восстановить на 4% больше паролей, чем dive.rule.

Для использования правила, необходимо сохранить файл, в директорию “rules”.

Далее правило можно вызвать как предустановленное:

```
hashcat.exe -m 1000 -a 0 -O -w 4 E:\hashs.txt E:\dicts\rockyou.txt -r
rules\OneRuleToRuleThemAll.rule
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: E:\hashs.txt
Time.Started.....: Mon Jan 04 18:37:50 2021 (3 mins, 10 secs)
Time.Estimated...: Mon Jan 04 18:41:00 2021 (0 secs)
Guess.Base.......: File (E:\dicts\rockyou.txt)
Guess.Mod........: Rules (rules/one.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3953.2 MH/s (36.95ms) @ Accel:64 Loops:256 Thr:1024 Vec:1
Recovered........: 11/20 (55.00%) Digests
Progress.........: 745836246080/745836246080 (100.00%)
Rejected.........: 340151290/745836246080 (0.05%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:51968-51995 Iteration:0-256
Candidates.#1....: $HEX[30383431393835] -> $HEX[042a0337c2a156616c6c732103]
Hardware.Mon.#1..: Temp: 63c Fan: 69% Util:100% Core:1873MHz Mem:3802MHz Bus:16

Started: Mon Jan 04 18:37:49 2021
Stopped: Mon Jan 04 18:41:01 2021
```

Перебор по словарю с правилами мутации

На тестовом стенде программе потребовалось 192 секунды на перебор 20-и хэшей по словарю, содержащему 745 миллиардов паролей. Данное правило позволило восстановить 11 паролей из 20, что на два меньше, чем в dive.rule, однако программа отработала в 2 раза быстрее, чем в случае с dive.rule.

Перебор по маске

Перебор по маске является разновидностью полного перебора.

В hashcat существуют предопределенные наборы символов, которые позволяют описывать маску для перебора.

```
 ? | Charset
===+=========
 l | abcdefghijklmnopqrstuvwxyz
 u | ABCDEFGHIJKLMNOPQRSTUVWXYZ
 d | 0123456789
 h | 0123456789abcdef
 H | 0123456789ABCDEF
 s |  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
 a | ?l?u?d?s
 b | 0x00 - 0xff
```

Предопределенные наборы символов

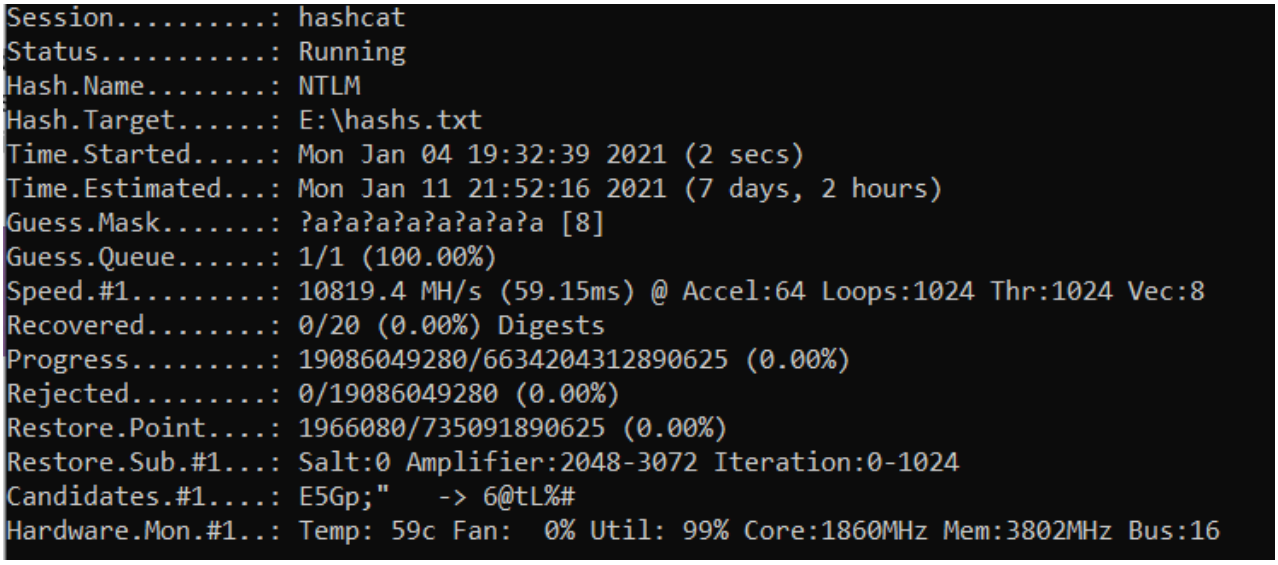
Например, маска `?a?a?a` означает, что будут перебираться все комбинации символов из
набора: `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'()*+,-./:;<=>?@[\]^_\{|}~` длиной 3. Однако набор символов ?a является

очень большим, что не позволяет в обозримое время перебирать длинные пароли. Например, для перебора 20 хэшей по маске `?a?a?a?a?a?a?a?a` (длина 8) на тестовом стенде потребуется около 7-и дней.

```
hashcat.exe -m 1000 -a 3 -O -w 4 E:\hashs.txt ?a?a?a?a?a?a?a?a
```

где:

- `a 3` — Режим перебора по маске.
- `?a?a?a?a?a?a?a?a` — Маска, означающая пароль, длиной 8, состоящий из символов `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 !"#$%&'(),-./:;<=>?@[\]^_{|}~+*`.



Перебор по маске

Перебором по маске с набором символов ?a можно пользоваться для перебора коротких паролей, например, длиной от 1 до 6. Для того чтобы не вводить команду 6 раз, можно воспользоваться режимом инкрементирования:

```
hashcat.exe -m 1000 -a 3 -O -w 4 E:\hashs.txt ?a?a?a?a?a?a --increment --
increment-min 1
```

где:

- `-increment` — Запуск в режиме инкрементирования.
- `-increment-min 1` — Минимальная длина пароля 1.

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: E:\hashs.txt
Time.Started.....: Mon Jan 04 19:56:01 2021 (1 min, 6 secs)
Time.Estimated...: Mon Jan 04 19:57:07 2021 (0 secs)
Guess.Mask.......: ?a?a?a?a?a?a [6]
Guess.Queue......: 6/6 (100.00%)
Speed.#1.........: 11260.0 MH/s (54.29ms) @ Accel:64 Loops:1024 Thr:1024 Vec:8
Recovered........: 2/20 (10.00%) Digests
Progress.........: 735091890625/735091890625 (100.00%)
Rejected.........: 0/735091890625 (0.00%)
Restore.Point....: 81450625/81450625 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:8192-9025 Iteration:0-1024
Candidates.#1....: 3v7Ej~ ->  ~ ~~~
Hardware.Mon.#1..: Temp: 65c Fan: 71% Util: 98% Core:1822MHz Mem:3802MHz Bus:16

Started: Mon Jan 04 19:55:57 2021
Stopped: Mon Jan 04 19:57:08 2021
```

Перебор по маске с инкрементом

Перебор 20 хэшей паролей с длиной от 1 до 6 символов был осуществлен за 71 секунду, а паролей длиной 7, был бы осуществлен уже за 2 часа. Попытка восстановления исходных хэш-значений паролей с длиной меньше 8 символов может быть полезна в сети AD, в том случае, если минимальная длина пароля была когда-либо установлена меньше чем восемь символов.

Перебор по маске с самодельными наборами

Для ускорения процесса перебора по маске, а также возможности перебора более длинных паролей, можно воспользоваться некоторыми правилами и закономерностями. Если рассмотреть описанные выше методы, то можно заметить, что большое количество полученных паролей начинается с большой буквы, после неё следуют несколько строчных букв, а потом цифры и/или спецсимволы. Это можно представить в виде маски, например:

```
?u?l?l?l?d?d?d?d
```

```
Session..........: hashcat
Status...........: Exhausted
Hash.Name........: NTLM
Hash.Target......: E:\hashs.txt
Time.Started.....: Mon Jan 04 20:24:15 2021 (1 sec)
Time.Estimated...: Mon Jan 04 20:24:16 2021 (0 secs)
Guess.Mask.......: ?u?l?l?l?d?d?d?d [8]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  7941.6 MH/s (40.94ms) @ Accel:64 Loops:676 Thr:1024 Vec:8
Recovered........: 1/20 (5.00%) Digests
Progress.........: 4569760000/4569760000 (100.00%)
Rejected.........: 0/4569760000 (0.00%)
Restore.Point....: 6760000/6760000 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-676 Iteration:0-676
Candidates.#1....: Madr4969 -> Xzqz9999
Hardware.Mon.#1..: Temp: 54c Fan: 65% Util: 99% Core:1885MHz Mem:3802MHz Bus:16

Started: Mon Jan 04 20:24:14 2021
Stopped: Mon Jan 04 20:24:16 2021
```

Перебор по маске

Существенно снизив набор символов, пароль длиной 8 символов можно перебрать за 2 секунды.

Перебор по наборам масок

Если мы хотим перебрать множество вариантов, состоящих из масок вида: большая буква, несколько строчных букв, цифры и/или спецсимволы, то придется запускать hashcat несколько десятков раз. Для упрощения задачи существуют hcmask-файлы — файлы набора масок. Создадим такой файл небольшим скриптом на Python:

```
for first in range(1,7):
    for digits in range(0,6):
        for second in range(0,3):
            mask = "?l3@17,!@#$%^&*.,?u"+str("?1"*first)+str("?d"*digits)+str("?
2"*second)
            if len(mask)<37 and len(mask)>32:
                print("?l3@17,!@#$%^&*.,?u"+str("?1"*first)+str("?
d"*digits)+str("?2"*second))
                print("?l3@17,!@#$%^&*.,"+str("?d"*digits)+"?u"+str("?
1"*first)+str("?2"*second))
```

Этот скрипт создаст 62 строки вида:

```
?l3@17,!@#$%^&*.,?u?1?1?d?d?d?d?2?2
```

Каждая из созданных строк состоит из трех полей, разделенных запятыми:

- Самодельный набор символов 1.
- Самодельный набор символов 2.
- Маска, состоящая из наборов символов.

Для перебора с использованием файла масок можно воспользоваться следующей командой:

```
hashcat.exe -m 1000 -a 3 -O -w 4 E:\hashs.txt E:\hashcat\mask.hcmask
```

где:

`E:\hashcat\mask.hcmask` — Путь к файлу масок.

Перебор 20 хэшей паролей по 62 маскам от 8 до 9 символов занял 44 минуты.

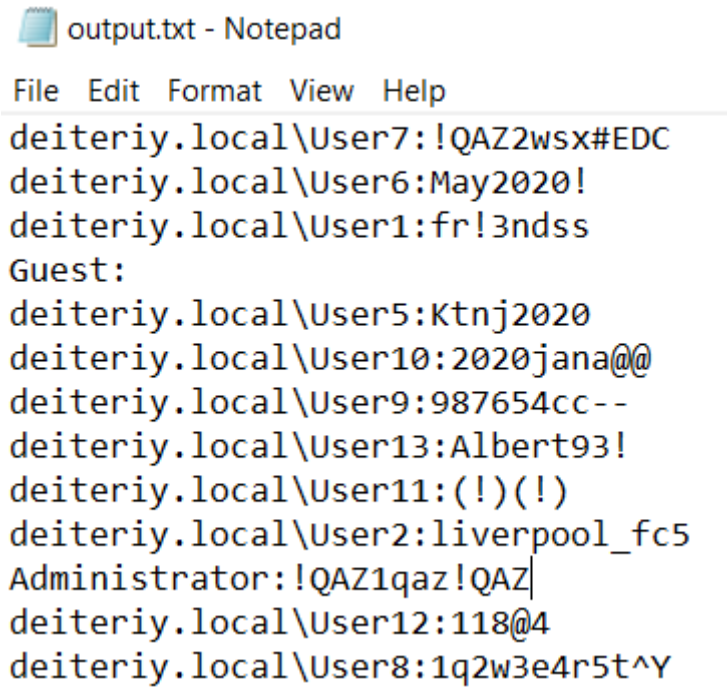
Как сделать удобный вывод

После всех операций выше, pot-файл будет заполнен полученными паролями и соответствующими им хэшами. Чтобы узнать какой пароль соответствует какому пользователю, можно выполнить следующую команду:

```
hashcat.exe -m 1000 --show --username --outfile E:\output.txt --outfile-format 2
E:\hashs.txt
```

где:

- `-username` — Игнорирование имя пользователя в хэш-файле.
- `-outfile E:\output.txt` — Файл, в который будет сохранен вывод программы.
- `-outfile-format 2` — Формат вывода программы, где 2 — в открытом виде.



Пример удобного вывода пар пользователь — пароль

Сравнение эффективности

Для тестирования скорости перебора и количества получаемых паролей каждым из описанных выше методов, бы произведен перебор хэшей, полученных из существующей организации. С контроллера домена было получено 7168 пользователей с 5267 уникальными хэшами.

▶ Методы перебора

Метод	Уникальных* хэшей восстановлено	Процент восстановленных хэшей	Затраченное время в секундах	Хэшей восстановлено
1	0	1.689766	4	89
2	0	3.721283	122	196
3	0	0.645529	2	34
4	0	19.66964	448	1036
5	137	30.20695	34784	1591
6	9	2.41124	92	127
7	0	18.54946	198	977
8	63	28.0805	15684	1479

9	1	1.898614	46	100
10	0	0.322764	73	17
11	71	7.404595	2668	390

*Уникальный хэш — хэш, который не был получен ни в одном из других методов.

Видим, что наибольшее число паролей восстановилось в методе “*5. Словарь realunique с правилами мутации dive.rule*”, который для получения конечных результатов требует 10 часов работы на тестовом стенде. В два раза быстрее отрабатывает метод “*8. Словарь realunique с правилами мутации OneRuleToRuleThemAll.rule*”, уступая предыдущему всего на 2% восстановленных паролей. Если требуется результат "как можно скорее", то можно воспользоваться методами “*4. Словарь rockyou.txt с правилами мутации dive.rule*” или “*7. Словарь rockyou.txt с правилами мутации OneRuleToRuleThemAll.rule*”, которые отличаются друг от друга на 1% хэшей и уступают предыдущим методам почти на 10%, но при этом, отрабатывают за 7 и 3,5 минуты соответственно.

Комбинации методов

Наилучшие результаты достигаются путем использования нескольких из описанных выше методов.

В файле ниже приведена полная таблица со всеми комбинациями.

https://www.dropbox.com/s/v4xpi4vr7knve1l/combo_1_2_3_4_5_top.xlsx?dl=0

Ниже переведены лучшие комбинации из наборов методов от 2 до 5.

Топ 7 наборов методов по количеству восстановленных хэшей

Набор методов №1

Комментарий: Лучший по количеству восстановленных хэшей
Метод 1: Словарь из календарных единиц с правилами мутации dive.rule
Метод 2: Словарь из календарных единиц с правилами мутации OneRuleToRuleThemAll.rule
Метод 3: Набор масок длиной от 7 до 9
Метод 4: Словарь realunique с правилами мутации OneRuleToRuleThemAll.rule
Метод 5: Словарь realunique с правилами мутации dive.rule

Количество уникальных хэшей: 658
Процент восстановленных хэшей: 34.53579
Затраченное время в секундах: 53320
Всего восстановлено: 1819

Набор методов №2

Комментарий: Лучший по количеству уникальных хэшей
Метод 1: Словарь rockyou.txt с правилами мутации dive.rule
Метод 2: Словарь rockyou.txt с правилами мутации OneRuleToRuleThemAll.rule
Метод 3: Набор масок длиной от 7 до 9
Метод 4: Словарь realunique с правилами мутации OneRuleToRuleThemAll.rule
Метод 5: Словарь realunique с правилами мутации dive.rule

Количество уникальных хэшей: 1498
Процент восстановленных хэшей: 33.60547
Затраченное время в секундах: 53782
Всего восстановлено: 1770

Набор методов №3

Комментарий: Высокая скорость и большое количество восстановленных хэшей #1
Метод 1: Словарь из календарных единиц с правилами мутации dive.rule
Метод 2: Словарь rockyou.txt с правилами мутации OneRuleToRuleThemAll.rule
Метод 3: Словарь из календарных единиц с правилами мутации OneRuleToRuleThemAll.rule
Метод 4: Набор масок длиной от 7 до 9
Метод 5: Словарь realunique с правилами мутации dive.rule

Количество уникальных хэшей: 263
Процент восстановленных хэшей: 33.33966
Затраченное время в секундах: 37834
Всего восстановлено: 1756

Набор методов №4

Комментарий: Лучший набор из двух методов
Метод 1: Словарь realunique с правилами мутации OneRuleToRuleThemAll.rule
Метод 2: Словарь realunique с правилами мутации dive.rule

Количество уникальных хэшей: 462
Процент восстановленных хэшей: 32.23847
Затраченное время в секундах: 50468
Всего восстановлено: 1698

Набор методов №5

Комментарий: Высокая скорость и большое количество хэшей #2
Метод 1: Словарь из календарных единиц с правилами мутации dive.rule
Метод 2: Словарь rockyou.txt с правилами мутации dive.rule
Метод 3: Словарь из календарных единиц с правилами мутации

OneRuleToRuleThemAll.rule
Метод 4: Набор масок длиной от 7 до 9
Метод 5: Словарь realunique с правилами мутации OneRuleToRuleThemAll.rule

Количество уникальных хэшей: 188
Процент восстановленных хэшей: 31.93469
Затраченное время в секундах: 18984
Всего восстановлено: 1682

Набор методов №6

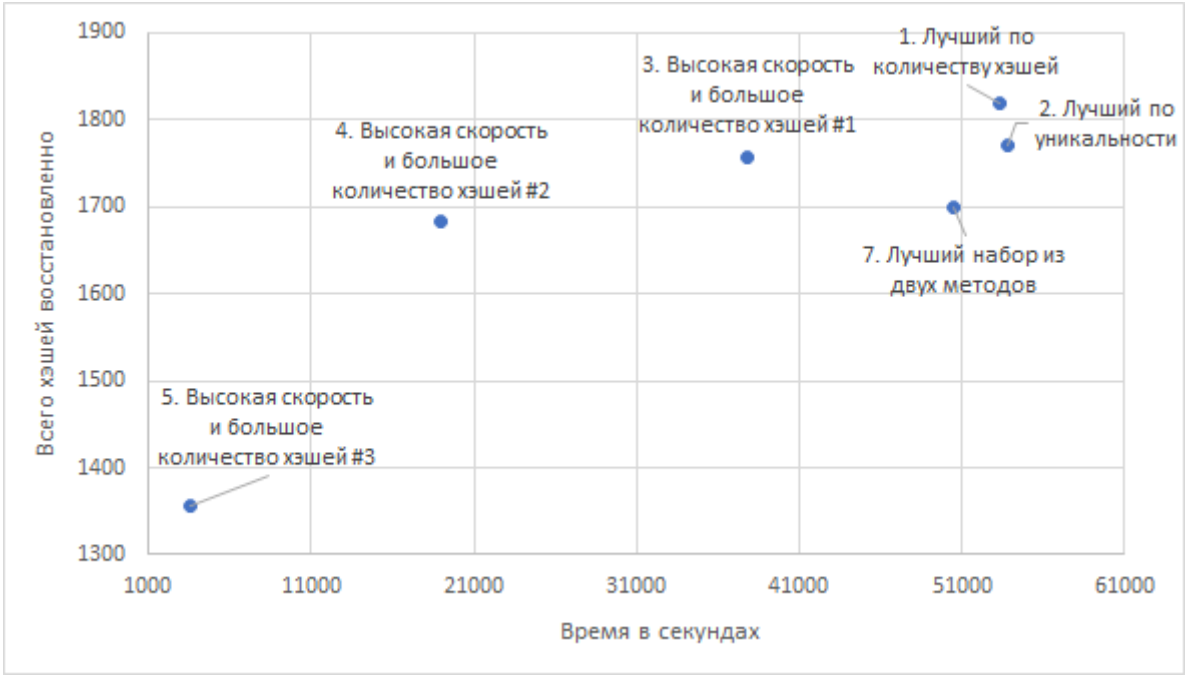
Комментарий: Высокая скорость и большое количество хэшей #3
Метод 1: Словарь realunique
Метод 2: Словарь из календарных единиц с правилами мутации dive.rule
Метод 3: Словарь rockyou.txt с правилами мутации dive.rule
Метод 4: Словарь rockyou.txt с правилами мутации OneRuleToRuleThemAll.rule
Метод 5: Набор масок длиной от 7 до 9

Количество уникальных хэшей: 80
Процент восстановленных хэшей: 25.74521
Затраченное время в секундах: 3528
Всего восстановлено: 1356

Набор методов №7

Комментарий: Высокая скорость и большое количество хэшей #4
Метод 1: Словарь realunique
Метод 2: Словарь из календарных единиц с правилами мутации dive.rule
Метод 3: Словарь rockyou.txt с правилами мутации dive.rule
Метод 4: Словарь rockyou.txt с правилами мутации OneRuleToRuleThemAll.rule

Количество уникальных хэшей: 9
Процент восстановленных хэшей: 23.10613
Затраченное время в секундах: 860
Всего восстановлено: 1217



Сравнение наборов методов

В зависимости от целей и располагаемого времени, рекомендуется выбрать один из наборов, из списка выше. Однако наилучший результат будет достигнут путем применения самостоятельно сгенерированных словарей, масок и правил.