Терия

Брутфорс — метод угадывания пароля (или ключа, используемого для шифрования), предполагающий систематический перебор всех возможных комбинаций символов до тех пор, пока не будет найдена правильная комбинация.

Брутфорс по сути своей является одной из базовых и самых лёгких для понимания атак. По сути, мы берём что-то, где нужен логин и пароль, и пытаемся пароль (или комбинацию *логин:пароль*) перебрать с помощью специальных инструментов или руками.

Главными **минусами** брутфорса являются:

- протяженность по времени;
- заметность при отслеживании сетевого трафика (множество сообщений о вводе неправильных учётных данных);
- лёгкость предотвращения его реализации.

Типы брутфорсов:

- атака полным перебором;
- атака по маске;
- атаки по словарю;
- гибридные атаки грубой силы;
- обратные атаки грубой силы;
- подстановка учётных данных.

Атака полным перебором
Злоумышленники пытаются логически угадать учётные данные без помощи программных инструментов или других средств. Они могут выявить очень простые пароли и *PIN*-коды.

Атака по маске
Злоумышленник, который что-то знает о шаблоне паролей, может использовать атаку по маске. Атака по маске уменьшает количество комбинаций метода грубой силы за счёт предположений или использования информации о формате пароля.

Атаки по словарю
При стандартной атаке злоумышленник выбирает цель и вводит возможные пароли для этого имени пользователя. Они известны как словарные атаки.
Атаки по словарю — это самый простой инструмент в атаках брутфорсом. Хотя сами по себе они не обязательно являются атаками методом грубой силы, они часто используются в качестве важного компонента для взлома паролей. Некоторые взломщики просматривают несокращённые словари и дополняют слова специальными символами и цифрами или используют специальные словари.

Гибридные атаки
Гибридная атака обычно сочетает атаки по словарю и перебор. Эти атаки используются для определения комбинированных паролей, в которых смешиваются общие слова со случайными символами. Пример такой атаки методом грубой силы может включать такие пароли, как *Petrov1993* или *Dominator228*.

Атаки обратным перебором
Как следует из названия, атака обратным перебором меняет стратегию атаки, начиная с известного пароля. Затем хакеры ищут миллионы имён пользователей, пока не найдут совпадение. Многие из этих преступников начинают с утечки паролей, которые доступны в Интернете из существующих утечек данных.

Подстановка учётных данных
Есть комбинация имени пользователя и пароля, которая работает для одного веб-сайта — злоумышленник

попробует её и на множестве других.

Брут может быть реализован в рамках множества протоколов приложений. Реализовывать брут можно в рамках *SSH*, *HTTP*, можно брутфорсить базы данных, пароли для почтовых приложений, *Skype*, *Teams* — вариантов огромное множество.

Здесь важно понимать несколько вещей:

1. **Следят ли за трафиком сети.** Если да, стоит ли этот брут обнаружения вас (как нарушителя) в сети.
2. **Стоит ли цель потраченных усилий.** С одной стороны, мы можем брутфорсить сервис с рандомными фотографиями сотрудников и в итоге, сбрутив учётку, получить доступ к паре ни к чему не обязывающих фотографий, а с другой — можем брутфорсить *CMS*, доступ в которую позволит нам подгружать файлы, в том числе файлы с полезной нагрузкой.
3. **Знаем ли мы что-то о парольных политиках цели.** Если мы хотим сбрутить учётку пользователя, однако знаем, что, например, в организации каждый месяц сотруднику генерируется новый пароль, состоящий из 16 символов, в которые могут входить заглавные-прописные буквы, цифры и спецсимволы, мы не сбрутим такое за какой-то разумный срок и лучше оставить эту идею. А если после десяти неудачных попыток учётка ещё и блокируется, тем более.

Как определить уязвимый к бруту протокол/сервис:

- **Если это протокол:** пробить в Интернете возможность брутфорса данного протокола. Заодно вы сразу увидите предлагаемый для таких целей инструментарий. Стоит парой неправильных вводов проверить, блокируется ли учётная запись/есть ли тайм-аут.
- **Если это сервис:** проверить на наличие блоков аккаунта после нескольких неправильных вводов, также проверить на блокировку *IP*-адреса. Посмотреть, есть ли тайм-аут после неправильных попыток, есть ли автоматическая разблокировка аккаунта после какого-то временного промежутка

Стоит упомянуть две его разновидности — онлайн- и офлайн-брут.

Онлайн-брут

Предполагает активное взаимодействие по сети между атакующим и протоколом/сервисом.

Онлайн-взлом паролей атакует компьютерную систему через интерфейс, который она представляет своим законным пользователям, пытаясь угадать учётные данные для входа. Например, злоумышленник может попытаться угадать учётные данные пользователя для страницы входа в веб-приложение, для *SSH* или *Telnet*-сервера или для сетевой службы, такой как облегчённый протокол доступа к каталогам (*LDAP*), один из почтовых протоколов (*SMTP*, *POP3* или *IMAP*), *FTP* и другие.

> Легко отслеживается в сети, множество техник для предотвращения, в разы медленнее офлайн-брута, однако не требует предварительного доступа в систему.

Офлайн-брут

Не предполагает активного взаимодействия по сети между атакующим и протоколом/сервисом.

Автономный взлом паролей — это попытка восстановить один или несколько паролей из файла хранилища паролей, который был восстановлен из целевой системы. Обычно это файл диспетчера учётных записей безопасности (*SAM*) в *Windows* или файл `/etc/shadow` в *Linux*.

В большинстве случаев для автономного взлома паролей требуется, чтобы злоумышленник уже получил права администратора `/root` в системе для получения доступа к механизму хранения.

Однако возможно, что хэши паролей также могли быть извлечены непосредственно из базы данных с помощью *SQL*-инъекции, незащищенного простого текстового файла на веб-сервере или какого-либо другого плохо защищённого источника.

> Сложно отследить (так как в основном работает с локальными зашифрованными паролями), в разы быстрее, чем онлайн-брут, однако требует предварительного доступа в систему (чтобы выгрузить хэши паролей/зашифрованные пароли).

Офлайн-взлом пароля, как и его онлайн-аналог, может использовать различные методы для подбора пароля. Атака полным перебором использует все возможные комбинации паролей, состоящие из заданного набора символов, вплоть до заданного размера пароля.

Например, атака методом грубой силы может попытаться взломать восьмизначный пароль, состоящий из всех 95 печатаемых символов *ASCII*. Это будет означать, что существует возможных комбинаций (95 x 95 x 95 x 95 x 95 x 95 x 95 x 95), или 6 634 204 312 890 625 (6,6 квадриллионов) паролей. Если предположить, что скорость составляет один миллион угадываний в секунду, то для взлома восьмизначного пароля с помощью атаки грубой силы потребуется около 210 лет.

Злоумышленник, который что-то знает о шаблоне паролей, может использовать **атаку по маске** и тем самым уменьшить количество возможных комбинаций. Например, мы знаем что:

1. Пароль состоит из восьми символов.
2. Первый символ — верхний регистр.
3. Следующие пять символов — нижний регистр.
4. Следующий символ — число.
5. Следующий символ — символ.

Количество возможных комбинаций: 26 x 26 x 26 x 26 x 26 x 26 x 10 x 34, или 105 031 363 840. При 1 000 000 комбинаций в секунду взлом этого пароля с помощью атаки по маске может занять до 1,2 дня. Сравните это с 210 годами, которые потребовались бы, чтобы взломать тот же пароль с помощью атаки грубой силы (когда было сделано никаких предположений о пароле)

Ещё один пример офлайн-атаки — **перебор хэшей**. Допустим, нам удалось выгрузить хэш пароля, нам необходимо его взломать. Об инструментарии, используемом для этого, мы поговорим в следующих уроках, а пока что обсудим принцип.

У нас есть **два варианта:**

1. Атака Rainbow Table.
2. Offline Password Cracking с использование *GPU*. При данном виде атаки, используя мощность графического процессора видеокарты, злоумышленники «на лету» генерируют хэши паролей (может быть как словарь, так и полный перебор по параметрам) и сравнивают с хэшем пароля, который хотят взломать. Если есть полное совпадение, пароль подобран.

Rainbow Table — это атака, которая использует словарь предварительно вычисленных хэшей для всех паролей заданного набора символов и размера. Атака по радужной таблице предотвращается с помощью соли или случайного фрагмента данных, добавляемого к паролю перед его хэшированием (которые обычно хранятся вместе с паролем, потому что это необходимо, когда сравниваемый пароль хэшируется).

Злоумышленник с радужной таблицей должен иметь таблицу для каждого значения соли (обычно 32 бита или более), и каждая радужная таблица может иметь размер в несколько терабайт даже для небольшого пароля, например из семи символов. Добавление соли останавливает атаку *Rainbow Table*, но ничего не делает против атаки *Offline Password Cracking* на *GPU*, поскольку хэши генерируются, добавляя соль, на лету.

Инструменты

Medusa

Medusa — это быстрый модульный брутфорсер. В настоящее время он имеет более 21 модуля, некоторые из которых: *CVS, FTP, HTTP, IMAP, MS-SQL, MySQL, NCP (NetWare), PcAnywhere, POP3, PostgreSQL, rexec, rlogin, rsh, SMB, SMTP (VRFY), SNMP, SSHv2, SVN, Telnet, VmAuthd, VNC* и общий модуль-оболочка.

Присмотревшись к *Medusa* поближе, мы можем выделить **три его ключевые особенности**:

1. **Параллельное тестирование на основе потоков.** Относится к возможности одновременного тестирования методом перебора нескольких хостов, пользователей или паролей.
2. **Гибкий ввод данных пользователем.** Относится к использованию информации о целевом пользователе (хост/пользователь/пароль), которую вы собрали на этапе сбора информации, в качестве входных данных, которые помогают *Medusa* выполнять более точечный брутфорс на цели.

3. Одной из самых интересных особенностей *Medusa* является его **модульный дизайн**. Интересная функция *Medusa* заключается в том, что каждый сервисный модуль существует как независимый файл `.mod`. Это означает, что не требуется никаких изменений в основном приложении, чтобы расширить список поддерживаемых служб для перебора.

Hydra

Hydra — это очень быстрый онлайн-инструмент для взлома паролей, который может выполнять быстрые словарные атаки против более чем 50 протоколов, включая *Telnet, RDP, SSH, FTP, HTTP, HTTPS, SMB*, нескольких баз данных и т. д.
Компания *THC (The Hackers Choice)* создала *Hydra* для исследователей и консультантов по безопасности, чтобы показать, насколько легко можно получить несанкционированный доступ к системе удалённо.

Hydra может использовать либо **атаку на основе словаря**, где вы предоставляете ему явный список слов, либо атаку перебора, которая будет пробовать каждую возможную комбинацию букв. Каждая из этих атак имеет свои преимущества и недостатки.

Атака по словарю будет использовать предварительно скомпилированный список слов — это ускорит процесс взлома с помощью грубой силы, потому что программа будет проходить только каждое слово в списке слов, но если слово отсутствует в указанном списке слов, атака не удастся.

При целенаправленной атаке против кого-то можно было бы использовать что-то вроде *CUPP (Common User Passwords Profiler)*, чтобы создать список слов, более специфичный для цели (день рождения, псевдоним, адрес, имя питомца и т. д.). Введите данные, которые вы знаете или которые можете узнать в социальных сетях, и *Hydra* создаст список слов на основе ваших входных данных.

Полный перебор будет взламывать пароль, пробуя все возможные комбинации пароля, например, он будет пробовать *aaaa*, затем *aaab*, *aaac*, *aaae*. Это значительно увеличивает время атаки, но снижает вероятность неудачи.

Hashcat/John the Ripper

Hashcat — это достаточно быстрый, эффективный и универсальный инструмент подбора хэшей паролей. При использовании в неопасных целях, например при тестировании на проникновение в собственную инфраструктуру, он может выявить скомпрометированные или легко угадываемые учётные данные.
Хакеры используют *Hashcat*, доступный для загрузки во всех основных операционных системах, для автоматизации атак на пароли. Это даёт пользователю возможность перебирать хранилища учётных данных с использованием известных хэшей, проводить атаки по словарю и радужные таблицы, а также преобразовывать читаемую информацию о поведении пользователя в атаки с использованием комбинации хэшированного пароля.

John the Ripper (JtR) — это инструмент для взлома паролей, изначально созданный для систем на базе *UNIX*. Он был разработан для проверки надёжности пароля, паролей, зашифрованных (хэшированных) методом перебора, и взлома паролей с помощью словарных атак

Если кратко, *John the Ripper* был настроен для работы с ЦП для взлома паролей, тогда как *Hashcat* (в первые дни его выпуска) был всего лишь инструментом для работы с графической вычислительной мощностью. Теперь разработчики заставили его работать и с ЦП, но за счёт снижения эффективности. Так что при работе с *CPU* (ЦП) выигрывает *John the Ripper*, но *Hashcat* выигрывает, когда дело доходит до взлома паролей с помощью графического процессора.

Механизмы защиты от брута

Блокировка IP-адреса

Блокировка бывает **ручной** и **автоматической**.

- **Ручная блокировка** *IP* отлично подходит, если вы знаете неправильный *IP*-адрес. Однако большинство атак методом перебора происходит с неизвестных вам *IP*-адресов. Здесь на помощь приходит автоматическая блокировка *IP*.
- При **автоматической блокировке** *IP*-адреса вы определяете количество неудачных попыток входа, которые могут произойти за определённый период времени (в секундах). Если в этот период количество неудачных попыток входа в систему больше установленного, *IP*-адрес, с которого производятся эти попытки, будет добавлен в список заблокированных *IP*-адресов. Таким образом, *IP*-адрес будет заблокирован.
  В автоматической блокировке *IP* вы также можете определить исключения: набор *IP* адресов, которые исключены из автоматического определения

Блокировка учётки при достаточном количестве неудачных попыток ввода

Самой встречающейся защитой от брутфорса, с которой вы встретитесь на практике, будет **блокировка учётки при достаточном количестве неудачных попыток ввода**.
При достижении определённого количества неудачных попыток ввода учётных данных аккаунт блокируется. В зависимости от настроек конкретной системы аккаунт может блокироваться или до разблокировки администратором, или до автоматической разблокировки по истечении какого-то времени.
Данный метод защиты применяется практически везде: в вебе почти в каждом сервисе с аутентификацией/авторизацией будет реализован данный метод защиты. По умолчанию 3-5 неправильных попыток приводят к блокировке учётки. Некоторые системы реализуют постепенные тайм-ауты: вначале при неправильном вводе вам дадут тайм-аут на 10 секунд, потом — 20 секунд, одна минута, 10 минут , 30 минут и окончательная блокировка.

Fail2ban

Любая служба, доступная в Интернете, уязвима для атак злоумышленников. Если ваша служба требует аутентификации, нелегитимные пользователи и боты попытаются проникнуть в вашу систему, неоднократно пытаясь аутентифицироваться с использованием разных учётных данных.
Типичным примером этого является *SSH*, который будет объектом атак ботов, пытающихся подобрать общие имена учётных записей. К счастью, такие сервисы, как **Fail2ban**, были созданы, чтобы помочь нам смягчить эти атаки.
Fail2ban работает, динамически изменяя правила брандмауэра, чтобы запретить адреса, которые безуспешно пытались войти в систему определённое количество раз.

Настройки защиты конкретных сервисов содержатся в файле /etc/fail2ban/jail.conf
Нас интересуют следующие **параметры**:

- `ignoreip` — *IP*–адреса, которые не должны быть заблокированы. Можно задать список *IP*-адресов, разделённых пробелами, маску подсети или имя *DNS*–сервера.
- `bantime` — время бана в секундах, по истечении которого *IP*–адрес удаляется из списка заблокированных.
- `maxretry` — количество подозрительных совпадений, после которых применяется правило. В контексте *[ssh]* — это число неудавшихся попыток логина, после которых происходит блокировка.
- `enabled` — значение *true* указывает, что данный *jail* активен, *false* выключает действие изолятора.
- `port` — указывает, на каком порте или портах запущен целевой сервис. Стандартный порт *SSH*–сервера — 22, или его буквенное наименование — *ssh*.
  Рекомендации по безопасности:

1. В ignoreip изменить стандартное значение 127.0.0.1/8 на другое, так как значение по умолчанию при компрометации одного из хостов приводит к тому, что с данного хоста можно запускать брутфорс по сетке без триггера *Fail2ban*.
2. Изменить стандартный *findtime* (определяет длительность интервала в секундах, за которое событие должно повториться определённое количество раз, после чего санкции вступят в силу) с пяти минут на полчаса-час чтобы усложнить работу «медленных» ботов.

Обход ограничений брутфорса

Тайм-аут брута до блокировки уз

Начнём с наиболее часто встречающейся техники защиты — **блокировка уз после нескольких неудачных попыток**.

Допустим, у нас есть сервис, мы знаем логин и хотим забрутить его форму авторизации. Мы знаем, что после пяти неправильных вводов аккаунт блокируется.

Также мы понимаем, что после четырёх попыток и 5-10-15 минут таймер обнуляется. Это проверяется вручную.

В таких условиях мы можем настроить брутфорс с тайм-аутом: четыре попытки — таймаут — четыре попытки и так до конца.

Данная техника реализуется редко, потому что, допустим, тайм-аут между обнулением количества неудавшихся попыток — примерно десять минут. У вас есть список из 200 паролей. Это 50 сессий, и после каждой последующей должно пройти четыре минуты. Следовательно, только на паузы будет потрачено 500 минут/8 часов 20 минут.

Поэтому главная проблема такого метода обхода — большая длительность его работы.

А 300 паролей — это, мягко говоря, несерьёзный словарь для брута (посмотрите тот же `rockyou.txt`).

Распределяем брут по учёткам (password spraying)

Поговорим о бруте по учёткам. Одну учётку блокируют после пяти неудачных попыток ввода, но **что если мы будем подставлять один пароль к пятидесяти учёткам?**

Смысл данной техники в том, что мы берём широкий спектр логинов (например, благодаря средствам *OSINT* мы поняли, что логин в компании — это *первая буква имени.фамилия*) и пробуем подставить в них один пароль. Обратная ситуация при брутфорсе пароля, только мы подбираем не пароль, а учётку под пароль. Это может быть легко реализовано через *Sniper*-режим атаки в *Burp Suite*.

Распределяем брут по IP-адресам

Допустим, мы знаем, что в рамках данного приложения практикуется бан *IP*-адреса при попытках брута. Нам нужно обойти данное ограничение. Для этого мы завернём трафик через *Tor*, и таким образом при блокировке одного *P*-адреса у нас уже будет другой для продолжения нашей атаки.

Распределяем брут по времени

Данная техника завязана на использовании оптимального количества неудачных попыток до блокировки аккаунта и выдерживании точного времени до обнуления таймера. После этого вновь делается оптимальное количество попыток, и так до получения нужного пароля.

Практика

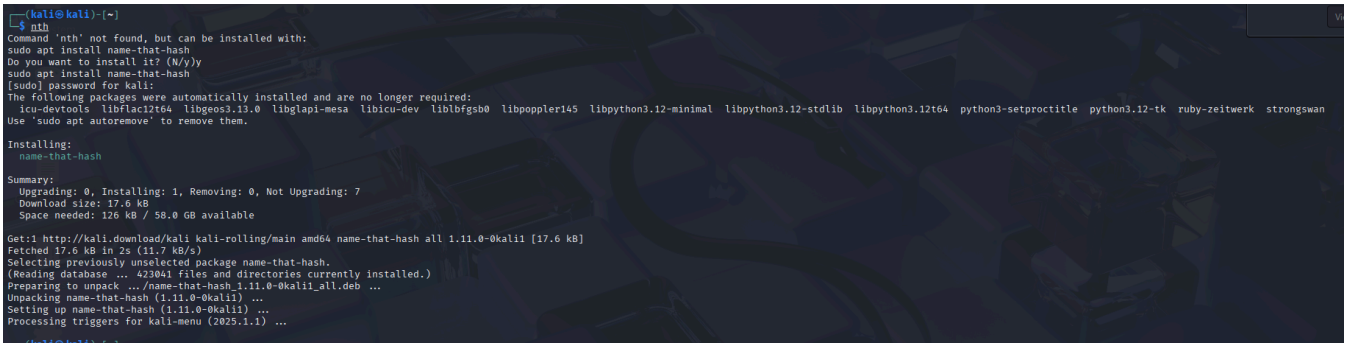
Вам дано несколько хэшей:

1. c7c06c1eb863fa23513834077a1ef3de11e3df81b45c8497974738739812543 0a53c9b5f9f84f0210819da784bf6cdb307c0dc1e019a6a9334e73990b947e3e9

2. b43f1d28a3dbf30070bf1ae7c88ee2784047fc86d7be8620c8510debbd8555b3ef0b96376a4dd494ae0561580274bcf7a3069f5c0beceff63d1237a13d4d72b7
3. 0a9b3f20d3d588b98616f5792d957709f62a5ddd2662436e46c752a5000db6877c8c58997bd0cadc3c8f14827510cab5df92170a8ceda6ecb8275fa6b5d8f375

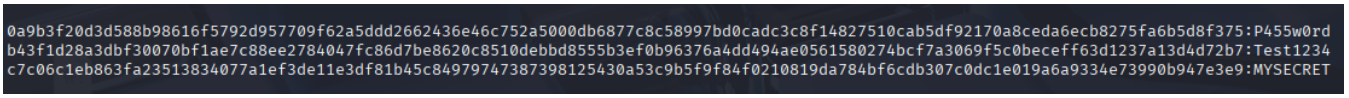
   Вам необходимо получить изначальный пароль, используя *Hashcat*. **Все три хэша должны быть сбрутфорсены одной командой.**

```
apt install nth
```




```
hashcat -h | grep "SHA"
Вероятно это SHA-512
1700 | SHA2-512                                                   | Raw Hash
```

```
hashcat -m 1700 -a 0 hash.txt /usr/share/wordlists/rockyou.txt
-m решим хеша , -a выбор режима, файл с хешами, путь к словарю
```




```
0a9b3f20d3d588b98616f5792d957709f62a5ddd2662436e46c752a5000db6877c8c58997bd0cadc3c8f14827510cab5df92170a8ceda6ecb8275fa6b5d8f375:P455w0rd
b43f1d28a3dbf30070bf1ae7c88ee2784047fc86d7be8620c8510debbd8555b3ef0b96376a4dd494ae0561580274bcf7a3069f5c0beceff63d1237a13d4d72b7:Test1234
c7c06c1eb863fa23513834077a1ef3de11e3df81b45c8497974738739812543 0a53c9b5f9f84f0210819da784bf6cdb307c0dc1e019a6a9334e73990b947e3e9:MYSECRET
```