Шифрование данных

Если вам нужно защитить данные, хранящиеся на твердотельном накопителе, жестком диске или любом другом устройстве хранения данных, вам нужно будет зашифровать все эти данные. Иногда это называют шифрованием неактивных данных. Это касается не только отдельных файлов, которые могут храниться на этих устройствах, но и всего, что находится на устройстве хранения данных, если используется шифрование на уровне всего диска или тома.

В операционной системе Windows для этого можно использовать BitLocker. Если вы используете Mac OS, то можете использовать FileVault. В других операционных системах есть другие способы шифрования всего содержимого на одном томе.

Вам также может потребоваться зашифровать отдельный файл, находящийся в системе, а не весь том. В Windows вы можете использовать EFS. Это расшифровывается как Шифрующая файловая система. Это шифрование на уровне файла, встроенное в файловую систему NTFS. И если вы используете Mac OS, Linux или даже Windows, существует множество сторонних утилит, которые могут выполнять аналогичную функцию.

В Windows вы можете посмотреть свойства файла или папки. В разделе «Дополнительные атрибуты» можно выбрать «Шифровать содержимое для защиты данных» и включить EFS. Большая часть данных, которые мы используем в Интернете, хранится в базах данных. И, конечно, существуют различные методы защиты данных в этих файлах баз данных.

Например, вы можете настроить прозрачное шифрование. При этом используется симметричный ключ для шифрования всех данных в базе. Вам нужно будет выполнять шифрование или дешифрование данных каждый раз, когда информация извлекается из базы. Некоторые данные в вашей базе могут не быть конфиденциальными. Таким образом, в базе могут быть как защищённые или зашифрованные данные, так и данные, которые по-прежнему доступны в виде обычного текста.

Вот пример таблицы в базе данных. Это база данных сотрудников, в которой указаны идентификационные номера сотрудников, их имена, фамилии и номера социального страхования. Конечно, вы можете зашифровать всю базу данных с помощью симметричного ключа, чтобы все эти данные были зашифрованы. Как видите, мы понятия не имеем, какая часть этих данных может быть связана с именем сотрудника, его идентификационным номером или номером социального страхования.

Но, конечно, для просмотра этой информации требуются дополнительные ресурсы. И каждый раз, когда нам нужно выполнить поиск по всей базе данных, нам фактически приходится расшифровывать все данные в этой базе. Один из способов избежать дополнительных затрат — шифровать только определенные типы данных в базе.

В этом примере мы выполняем шифрование на уровне столбца, при котором идентификатор сотрудника, имя и фамилия отображаются в виде обычного текста. Если вам нужно найти имя или идентификатор, вы можете сделать это очень быстро, не расшифровывая другие типы данных. Но если вам нужен доступ к номеру социального страхования человека, вам придётся расшифровать либо весь столбец, либо одну конкретную запись, чтобы получить доступ к этим данным.

Ещё одно распространённое место, где используется шифрование, — это отправка данных по сети. Мы хотим быть уверены, что всё, что мы передаём между двумя

устройствами, защищено. И если кто-то подключится к этому соединению и просмотрит эти данные, он не сможет разобраться в них. Например, сейчас вы, скорее всего, смотрите это видео в браузере. И все коммуникации, происходящие в вашем браузере, скорее всего, осуществляются по протоколу HTTPS, а это значит, что всё, что передаётся по сети, зашифровано.

Если вам необходимо подключить разные сайты друг к другу или подключить отдельных пользователей для удаленного доступа, мы обычно используем VPN для обеспечения такого шифрования. Это расшифровывается как виртуальная частная сеть. И это эффективно создает зашифрованный туннель, по которому вы можете отправлять всю информацию в туннель на другую сторону. И все, что находится внутри этого туннеля, будет зашифровано.

Обычно это используется в клиентских VPN-сетях с протоколом SSL или TLS. А если вы соединяете два сайта, то для обеспечения VPN-подключения мы обычно используем IPsec. Чтобы шифрование и дешифрование были успешными, обе стороны должны использовать одни и те же алгоритмы шифрования. Эта формула используется не только для шифрования, но и для расшифровки данных на другой стороне.

Как правило, обе стороны с самого начала договариваются об использовании одного или нескольких алгоритмов шифрования, чтобы обе стороны точно знали, чего ожидать при получении информации. Зачастую конечный пользователь не видит подробностей используемых алгоритмов. Но он знает, что использует определённое приложение. И он хочет быть уверен, что другая сторона использует аналогичное приложение, чтобы процессы шифрования и дешифрования были совместимы.

Очевидно, что в зависимости от используемого алгоритма шифрования есть свои преимущества и недостатки. Некоторые алгоритмы обеспечивают более высокий уровень безопасности, другие работают быстрее, а третьи имеют более сложный метод реализации. Но как только обе стороны договариваются о приложении, которое будет использоваться для шифрования и дешифрования, всё остальное, как правило, происходит автоматически. Обычно администратор системы безопасности хорошо представляет себе требования пользователей. И он следит за тем, чтобы использовались подходящие алгоритмы шифрования.

Вот хороший пример того, почему так важно, чтобы обе стороны в разговоре использовали один и тот же алгоритм шифрования. Это очень широкие сравнения между алгоритмом шифрования DES и алгоритмом шифрования AES. Они обозначают стандарт шифрования данных и Расширенный стандарт шифрования. Вам не нужно знать особенности этих блок-схем для сдачи экзамена Security +. Но вы можете наглядно увидеть, что между обоими этими алгоритмами существует довольно много различий.

Алгоритм шифрования DES состоит из пяти этапов, которые включают в себя разделение данных на левый и правый незашифрованные тексты для получения в итоге 64-битного зашифрованного текста. Как видите, алгоритм AES работает немного иначе: вы берёте незашифрованный текст и секретный ключ, добавляете их к шифру и в итоге получаете зашифрованный текст. Существуют также разные версии AES, которые могут выдавать разные результаты. Очевидно, что вы не сможете зашифровать текст с помощью DES, а затем каким-то образом расшифровать его с помощью AES. Вы должны быть уверены, что используете совместимые алгоритмы шифрования и дешифрования с обеих сторон.

Ещё одна интересная особенность алгоритмов шифрования заключается в том, что мы точно знаем, как они работают. Сами алгоритмы обычно публикуются. Вы можете прочитать код или изучить математические выкладки и увидеть, как именно происходит процесс. Алгоритм обычно хорошо известен. На самом деле это делает алгоритм более надёжным, потому что мы можем видеть математические выкладки и процесс, который используется для создания шифрования.

Единственная важная информация, которой у нас нет, — это ключ. И хотя мы знаем, как работает алгоритм, мы всё равно не сможем ничего реконструировать, пока у нас нет этого ключа. Это очень похоже на работу дверного замка.

Мы знаем, как работают дверные замки. Мы знаем, как производить дверные замки. Мы знаем, что происходит внутри дверного замка, когда вы вставляете ключ. Но одного знания недостаточно, чтобы открыть запертую дверь. Вам нужен подходящий ключ, как и в случае с шифрованием и дешифрованием.

Этот ключ помогает определить конечный результат. Если вы шифруете, хешируете данные или создаёте цифровую подпись, всё это основано на этом ключе. И даже если у вас есть алгоритм и вы разбираетесь в математике, вам всё равно нужен ключ, чтобы получить доступ к данным. Вот почему мы всегда советуем хранить приватные ключи в тайне. Если кто-то получит доступ к вашему ключу, он сможет использовать его для взлома вашего замка. И теперь у него есть доступ ко всем вашим данным.

Как и всё остальное, ваши ключи шифрования и дешифрования подвержены атакам методом перебора. Это значит, что злоумышленник может перепробовать все возможные варианты, чтобы определить, каким может быть открытый или закрытый ключ. Мы можем эффективно предотвратить такие атаки методом перебора, создав очень, очень длинный ключ. В мире шифрования симметричный ключ длиной 128 бит или больше был бы очень распространённым и хорошо защищённым. Со временем, когда наши процессоры станут мощнее и мы сможем объединять множество различных процессоров, мы сможем увеличить размер ключей, чтобы их было сложнее подобрать методом перебора.

Это увеличение длины ключа применимо и к асимметричному шифрованию. Несмотря на то, что асимметричный ключ использует сложные математические операции с очень большими простыми числами, злоумышленник всё равно может подобрать его методом перебора. Нередко встречаются асимметричные ключи длиной 3072 бита и даже больше.

Это значит, что со временем нам, возможно, придётся создавать всё более и более крупные ключи, чтобы успевать за развитием технологий. Но есть и другие способы сделать существующие ключи намного более безопасными. Один из таких способов — многократное шифрование одного типа данных.

Например, вы можете хешировать пароль, затем хешировать хеш этого пароля, затем хешировать хеш хеша этого пароля и так далее. Это называется растяжением или усилением ключа. Это означает, что если кто-то захочет методом перебора расшифровать данные, которые были зашифрованы несколько раз с помощью этого метода растяжения ключа, ему придётся расшифровывать их несколько раз, чтобы понять, был ли метод перебора успешным. Это создаёт дополнительную нагрузку и, безусловно, увеличивает время процесса перебора.