УДК 004

БЕЗОПАСНОСТЬ ЛОКАЛЬНЫХ БАЗ ДАННЫХ НА ПРИМЕРЕ SQL SERVER COMPACT

Нафикова А.Р.

Доцент, кандидат физико-математических наук

Стерлитамакский филиал БашГУ,

Стерлитамак, Россия

Садыкова А.Р.

Студент,

Стерлитамакский филиал БашГУ,

Стерлитамак, Россия

Аннотация. Рассмотрен вопрос защиты локальной базы данных от несанкционированного доступа. Приведено сравнение того, как избежать хранения конфиденциальной информации в форме, допускающей несанкционированный доступ к данным. Особое внимание следует уделить шифрование файлов конфигурации, содержащих информацию о подключении к источнику данных. Приложение, разработанное для требований безопасности локальной базы данных в отношении защиты уровня выявлено на основе исследования.

Ключевые слова: безопасность; базы данных; SQL Server Compact.

SECURITY OF LOCAL DATABASES ON THE EXAMPLE OF SQL SERVER COMPACT

Nafikova A.R.

Associate Professor, Candidate of Physical and Mathematical Sciences

Sterlitamak branch of Bashkir State University,

Sterlitamak, Russia

Sadikova A.R.

Student,

Sterlitamak branch of Bashkir State University,
Sterlitamak, Russia

Annotation: This article highlights some of the criteria that information application developers can use to choose between a database solution or data warehousing.

Keywords: security; databases; SQL Server Compact.

Базы данных в настоящее время являются ключевым компонентом большинства клиент-серверных приложений. В дополнение к центральной базе данных на сервере часто используется локальная база данных клиента. Такой подход улучшает полную или частичную автономность приложения и повышает скорость обработки, поскольку клиентское приложение отвечает за большую часть данных, необходимых для работы, и устраняет необходимость доступа к удаленной базе данных. Однако, поскольку клиентские приложения могут хранить конфиденциальную информацию, обеспечение безопасности локальной базы данных является одной из наиболее важных задач при разработке приложений такого типа. Безопасность базы данных - это защита конфиденциальной информации, содержащейся в ней. Чтобы обеспечить безопасность базы данных, необходимо защитить базу данных от несанкционированного доступа и вредоносных или случайных изменений данных [1].

Самый простой способ защитить базу данных – установить пароль для подключения к базе данных. Для SQL Server Compact (реляционная база данных Майкрософт) шифрование файлов базы данных включается автоматически при установке пароля.

Однако по умолчанию строка подключения, которая содержит всю информацию, необходимую для подключения к базе данных, сохраняется в файле конфигурации. Как правило, этот файл находится в исполняемой папке

приложения и может быть открыт и прочитан любым пользователем. Эта ситуация неприемлема для приложений, на которые распространяются требования безопасности. Есть несколько способов избежать хранения конфиденциальных данных в виде простого текста:

1. Сохранить или сгенерировать пароль в коде приложения. В этом случае пароль не сохраняется в файле конфигурации, а заменяется непосредственно из кода путем изменения строки подключения к базе данных. Способ изменения строки подключения зависит от поставщика, выбранного для SQL Server Compact.

Однако такой подход к решению проблем безопасности имеет несколько недостатков. Во-первых, все копии клиентского приложения имеют одинаковый пароль. А во-вторых, исходный код приложения можно просмотреть с помощью специальной программы – дизассемблера. Поэтому злоумышленник может легко подключиться к локальной базе данных, открыв исполняемый файл или библиотеку, содержащую пароль для базы данных, с помощью дизассемблера.

2. Получить пароль для подключения к локальной базе данных с сервера. Этот метод защиты похож на предыдущий, но пароль (или метод его генерации) не хранится в открытом тексте в коде приложения. Кроме того, можно создать уникальный пароль для каждой клиентской базы данных в зависимости от параметров рабочего места клиента и т. д. При реализации этого метода желательно использовать криптографический защищенный канал связи между клиентом и сервером, так как в противном случае пароль может быть перехваченным и прочитанным во время передачи.

К недостаткам этого подхода относится тот факт, что вход в приложение возможен только при наличии соединения с сервером. Хотя это

накладывает некоторые ограничения на автономность приложения, оно не мешает работе с нестабильными соединениями.

3. Зашифровать раздел с помощью строки подключения в файле конфигурации. Методы защиты конфигурации могут использоваться для шифрования конфиденциальной информации, такой как имена пользователей и пароли в файлах конфигурации базы данных, строках подключения к базе данных и ключах шифрования. В файле конфигурации, в котором значения строки подключения шифруются с использованием метода защиты конфигурации, строка подключения не отображается в виде простого текста, но сохраняется в зашифрованном виде.

Шифрование и дешифрование содержимого конфигурационного файла выполняется с помощью класса, унаследованного от ProtectedConfigurationProvider.

.NetFramework включает надежного поставщика конфигурации по имени DpapiProtectedConfigurationProvider и RsaProtectedConfigurationProvider. Оба поставщика обеспечивают надежное шифрование данных, но если планируется использовать файлы конфигурации, зашифрованные на нескольких компьютерах, следует использовать класс RsaProtectedConfigurationProvider.

Класс DpapiProtectedConfigurationProvider использует интерфейс прикладного программирования защиты данных (DPAPI), который представляет собой криптографический интерфейс программирования для приложений операционной системы Windows, которые защищают конфиденциальность данных путем шифрования данных. Этот провайдер может быть настроен для защиты с помощью учетной записи компьютера или учетной записи пользователя.

Класс RsaProtectedConfigurationProvider использует функциональные возможности шифрования, предоставляемые классами RSA (сокращения Rivest, Shamir и Adleman). Это обеспечивает реализацию ключа RSA. RSA - это криптографический алгоритм, надежность которого основана на сложности факторизации больших чисел и вычисления дискретных логарифмов. Этот алгоритм шифрования классифицируется как асимметричный и использует два простых случайных числа для генерации открытого и закрытого ключей [2].

С платформой .NET Framework можно использовать своего собственного поставщика безопасной конфигурации. Если нужно использовать другой алгоритм шифрования, чем тот, который предоставляется классами DpapiProtectedConfigurationProvider и RsaProtectedConstructionProvider, рекомендуется реализовать собственный поставщик.

4. Как только клиент запускает приложение, он запрашивает пароль из базы данных. Хотя этот метод позволяет избежать сохранения пароля в файле конфигурации, можно подключиться к базе данных не только через клиентское приложение, но и с помощью сторонних программ. В этом случае пользователи могут напрямую подключаться к локальной базе данных и использовать клиентское приложение для доступа к скрытой от них конфиденциальной информации.

Вывод

Эти методы обеспечивают различные уровни защиты локальных баз данных. Метод следует выбирать исходя из требований безопасности к приложению, так как с повышением уровня защиты возрастает и стоимость реализации алгоритма.

Библиографический список:

1. Башарат И. Безопасность и шифрование баз данных: обзорное исследование/ Башарат И., азам Ф., Музафар А. В. / / Международный журнал компьютерных приложений. 2012. V. 47. № 12. С. 28-34.
2. Сингх Г.Суприя. Изучение алгоритмов шифрования (RSA, как DES, 3DES и AES) для обеспечения информационной безопасности // Международный журнал прикладных компьютерных программ. 2013. V. 67. № 19. С. 33-38

Оригинальность 81%