Курс «Защита информации»
кафедры радиотехники Московского физико-технического института (ГУ)

Эссе на тему
«Шифрование в базах данных»

выполнил студент 211 группы
© Дмитрий Омельченко

Содержание

Введение

Современные приложения масштаба предприятия имеют многоуровневую структуру. Распространенным примером являются приложения, построенные по трёхуровневой архитектуре. Такие приложения состоят из трёх обособленных частей, взаимодействующих друг с другом, причем каждая из частей работает на отдельном физическом устройстве и выполняет определенную функцию. Части трёхуровневого приложения – это клиент (клиентское приложение), сервер приложений и хранилище данных (например, реляционная база данных).

Клиентское приложение предназначено для взаимодействия с пользователем, отображения информации в удобном виде, получения от пользователя команд и перенаправления этих команд на сервер приложений для их исполнения. В качестве клиента может выступать программа, поставляемая разработчиком приложения, специфическая для конкретного приложения. Выделяют также класс так называемых «тонких» клиентов – клиентских приложений, не завязанных на логику конкретного приложения. В качестве такого клиента может выступать, например, интернет-браузер.

На сервере приложений реализуется логика приложений: сюда поступают запросы от всех клиентов, здесь они обрабатываются и клиентам возвращается результат обработки запросов. Для получения и хранения данных сервер приложений обращается к СУБД – системе управления базой данных. Данные в реляционных базах данных хранятся в виде таблиц, которые могут быть связаны между собой отношениями. Каждая таблица имеет набор полей различных типов – для хранения данных различных типов. СУБД предоставляет серверу приложений определенный интерфейс, для манипуляций со структурой базы и самими данными – например, язык SQL.

Все данные в приложении, построенном по трёхуровневой архитектуре, хранятся в базе данных. Сохранность и конфиденциальность этих данных – вопрос критической важности. В базе данных могут храниться конфиденциальные данные клиентов, которым приложение предоставляет услуги, например, адреса, телефоны, номера кредитных карт, состояние счетов и т.д. Если же речь идет о внутренней системе какой-либо организации, то примером таких данных могут быть величины ставок зарплат, данные о финансовых операциях организации и т.п.

Необходимо предпринимать самые серьёзные меры для предотвращения несанкционированного доступа и разглашения данных, которыми управляет приложение. Эти меры должны включать:

- защиту сети организации, в которой работает распределенное приложение, от атак извне;
- защиту базы данных от посягательств со стороны злоумышленных сотрудников самой организации;
- защиту данных, пересылаемых по сети между компонентами распределенного приложения (например, между сервером базы данных и сервером приложений) от прослушивания злоумышленными сотрудниками.

Первая из перечисленных видов угроз приводит к необходимости использования межсетевых экранов (брандмауэров) для закрытия доступа извне к внутренним серверам и ресурсам организации. На методах защиты от угроз второго и отчасти третьего типа мы остановимся более подробно.

Защита базы данных.

Защита данных, хранящихся в базе, включает в себя следующие аспекты:

- шифрование данных и вытекающая отсюда необходимость безопасного управления ключами;
- управление правами доступа к объектам базы данных.

Само по себе шифрование данных нисколько не увеличит безопасность , если не разработана продуманная система ролей, прав доступа пользователей к данным. Каждый пользователь должен иметь набор прав, минимально необходимый для выполнения его работы, и это должно быть реализованно исключительно посредством механизма управления правами доступа, но никак не шифрованием данных. Рассмотрим пример. Рядовой сотрудник какой-либо организации должен иметь доступ только к своим персональным данным. У него не должно быть возможности просматривать и редактировать данные других сотрудников. У руководителя, помимо доступа к своим персональным данным, должен также быть доступ к данным его подчиненных. Кроме того, у сотрудника отдела кадров должен быть доступ к данным множества сотрудников, помимо своих. Перечисленные требования могут быть реализованы путём шифрования данных сотрудников и распространения ключей для расшифрования только среди тех людей, которые должны иметь возможность доступа. К примеру, ключ для расшифрования данных рядового сотрудника должен быть у него самого, его руководителя и у сотрудника отдела кадров. Подобное использование шифрования данных является пагубным и нисколько не добавляет безопасности системе, поскольку встаёт задача сохранения ключей для расшифрования в секрете между несколькими пользователями системы, что потенциально представляет из себя не меньшую опасность. Задача ограничения доступа пользователей к данным должна быть реализована исключительно путём создания и назначения пользователям соответствующих ролей. Шифрование данных для этих целей использовать нельзя.

Использование шифрования базы данных целесообразно только при условии строгого контроля над распространением криптографических ключей: узнав ключ, злоумышленник сразу может расшифровать секретные данные, к которым у него есть доступ. В случае если ключи храняться отдельно от зашифрованных данных, шифрование защищает от простого воровства данных. Например, если будет похищена резервная копия содержимого базы, данные все равно не будут раскрыты.

Разработка системы шифрования базы данных требует рассмотрения следующих вопросов:

- определение частей данных, шифрование которых целесообразно;
- определение места, где будет происходить зашифрование и расшифрование данных;
- определение места, где будут храниться ключи;
- разработка механизма смены ключей по истечению определённого срока давности и перешифрования данных новыми ключами.

Какие данные шифровать?

Об этом следует задуматься в первую очередь. Шифрование абсолютно всех данных в базе отнюдь не обеспечивает совершенную их защищенность. Кроме того, следует иметь в виду, что зашифрование и расшифрование – ресурсоёмкие операции, особенно для некоторых систем шифрования (например, DES, 3DES и др.). Если все данные в базе хранятся в зашифрованном виде, то при исполнении каждого запроса на добавление, модификацию или выборку данных неизбежно выполняются процедуры зашифрования/расшифрования. Это создаёт дополнительную нагрузку на вычислительные мощности СУБД и может существенно понизить производительность всего приложения

вплоть до неприемлемо низкого уровня. При определении набора данных, подлежащих шифрованию, нужно стремиться к компромиссу между количеством скрытых данных, достаточным для обеспечения требуемого уровня безопасности, с одной стороны и приемлемым снижением производительности приложения с другой.

Управление ключами

Первое, на что следует обратить внимание – это генератор ключей. Стойкость криптосистемы напрямую зависит от того, насколько случайно выбираются ключи для шифрования. Следует убедиться в пригодности генератора ключей, предоставляемого СУБД.

Для уменьшения степени уязвимости может быть целесообразным использование нескольких ключей для шифрования данных в базе: если ключ только один, то попадание его в руки злоумышленников сразу скомпрометирует все данные. Если же ключей несколько, то раскрытой окажется только часть данных. Однако с увеличением количества ключей усложняется механизм шифрования, и становится труднее следить за распространением ключей только среди людей, которые должны иметь возможность расшифрования.

Время от времени ключи необходимо менять. Процедура смены ключей начинается с расшифрования базы данных старыми ключами, а затем её зашифрования новыми. Важно, что в течение этого процесса система хранения данных может быть недоступной для исполнения каких-либо запросов, и соответственно, сделать недоступным все распределенное приложение. Необходимо запланировать регулярное техническое обслуживание системы для смены ключей. Если же нельзя допустить временной недоступности приложения, то возможно применение методов зеркалирования базы данных: приложение работает на одной копии в режиме, допускающем только чтение данных, но не их модификацию, другая копия в это время перешифровывается новыми ключами, а затем происходит переключение приложения на перешифрованную базу.

Критическую важность имеет вопрос о месте хранения ключей, это непосредственно влияет на их безопасность. Можно хранить ключи в отдельной таблице или в каком-либо другом виде (например, в виде файла) на сервере базы данных. Но тогда администратору сервера, имеющему доступ ко всем данным на сервере, ключи легко доступны. Доверительная важность роли администратора резко возрастает, безопасность всех данных находится в руках одного человека. Более разумным представляется хранение ключей на носителе, управляемом исключительно аппаратно, но не программно. Механизм доступа системы шифрования также следует реализовать на аппаратном уровне, чтобы ключи никогда не покидали этого уровня, и потому были недоступны ни администратору ни разработчику системы.

Место шифрования данных

Можно выделить два подхода:
1). шифрование происходит на сервере базы данных средствами СУБД;
2). шифрование происходит вне сервера базы данных.

Шифрование на сервере базы данных имеет неоспоримое преимущество – оно полностью прозрачно для остальных компонентов приложения. Включение шифрования в этом случае не требует никаких изменений других компонентов. Данные поступают на сервер базы данных в незашифрованном виде и шифруются перед сохранением их на диск. При запросе на выборку данных они сначала расшифровываются на сервере, а затем отправляются клиенту. Таким образом никакие части системы кроме базы данных не замечают шифрования.

Список недостатков такого подхода впечатляет:

- большая дополнительная вычислительная нагрузка на сервер базы даных, способная существенно понизить его производительность;
- используемая СУБД должна поддерживать возможность шифрования, причем некоторые СУБД имеют ограничения: например, невозможность шифрования только части данных, либо шифруется вся база целиком, либо не шифруется ничего;
- зависимость от алгоритмов шифрования, встроенных в СУБД: можно использовать только те алгоритмы, которые реализованы в конкретной СУБД;
- способ хранения ключей может быть уязвим: если они хранятся в отдельной таблице, администраторы БД имеют неограниченный доступ к ключам, а значит и ко всем зашифрованным данным;
- между компонентами распределенного приложения данные пересылаются в открытом виде, поэтому если они не защищены от прослушивания, то шифрование данных в базе теряет смысл; необходимо использовать защищенные соединения.

Альтернативой вышеописанному подходу является вынесение шифрования за пределы сервера базы данных. Можно шифровать данные на стороне клиента. Тогда отпадает необходимость безопасных соединений, но требуется переработка клиента и сервера приложений. Безопасное управление ключами становится очень сложной задачей, поскольку ключи для расшифрования должны быть доступны многим клиентам.

Более предпочтительным представляется вынесение функций шифрования на отдельный сервер – сервер шифрования. Для выполнения какой-либо операции с данными приложение, работающее на сервере приложений, посылает запрос серверу шифрования (для этого должен быть разработан протокол общения приложения и сервера шифрования). Сервер шифрования производит необходимые операции по шифрованию данных из этого запроса и выполняет SQL-запрос к базе данных. Между базой и сервером шифрования данные пересылаются исключительно в зашифрованном виде, что снимает необходимость использования защищенного соединения между этими двумя компонентами приложения.

Данный подход особенно удобен, когда приложение использует не одну, а несколько баз данных. Среди других преимуществ:

- отделённость хранилища ключей от места хранения зашифрованных данных: доступ администратора базы данных к ключам закрыт;
- отсутствие дополнительной нагрузки на сервер базы данных;
- независимость от набора алгоритмов шифрования, предоставляемых СУБД: на сервере шифрования можно использовать любые алгоритмы сторонних производителей;
- неограниченные возможности для настройки системы шифрования под нужды конкретного приложения, под потитику безопасности организации;
- экономическая эффективность: сервер шифрования предоставляет унифицированные услуги по шифрованию данных, которыми могут пользоваться сразу несколько приложений.

Необходимость администрирования большего количества серверов и изменения компонентов приложения не портит общей картины привлекательности решения.

Общие замечания

Нельзя упускать из внимания побочные эффекты, возникающие от шифрования данных.

Зачастую зашифрованные данные занимают больше дискового пространства нежели незашифрованные. Например, при использовании DES (64-bit) в качестве системы шифрования значение типа Integer (4 байта) зашифровывается в 8-битную символьную строку. После зашифрования пространство, занимаемое данными, может увеличиться в 2-

3 раза. Перед шифрованием нужно позаботиться об обеспечении достаточного количества свободного места на сервере. Кроме того, шифрование может привести к изменению типа данных: зашифрованные данные не смогут храниться в прежних полях таблиц, поскольку их тип отличен от типа исходных данных (например, данные какого-либо числового типа могут зашифроваться в данные символьного типа). Изменение типа и размера данных специфично для каждого алгоритма шифрования. Нельзя упускать из внимания необходимость корректировки типов и размеров полей таблиц базы данных.

Шифрование индексированных полей таблиц сводит на нет ускорение поиска данных: порядок сортировки зашифрованных данных никак не будет соответствовать порядку исходных данных. При проектировании системы шифрования следует избегать шифрования индексированных полей. Если же избежать этого не удаётся, то разыскиваемое значение сначала нужно зашифровать и затем производить поиск среди зашифрованных данных. Это подразумевает внесение изменений в процедуру поиска данных.

Заключение

Современные стандарты безопасности информационных технологий для бизнеса и управления требуют уделять большое внимание сохранности данных. Шифрование данных, несомненно, обладает потенциалом повышения безопасности информационных систем. Однако оно может быть полезным только при условии продуманной экспертной проработки всей системы безопасности в целом.

Список литературы

1. Database Encryption in Oracle9i™, An Oracle Technical White Paper, February 2001
2. Oracle Advanced Security, Oracle data sheet
3. Securing Data at Rest: Developing a Database Encryption Strategy, A white Paper for Developers, e-Business Managers and IT, RSA Security Inc.
4. Don MacVittie, Time is Right for Database Encryption, Dec 9, 2003
5. Ulf Mattsson, Encryption of Enterprise Databases
6. Arup Nanda, Transparent Data Encryption, Oracle Magazine, September-October 2005