Содержание:

Понятие защиты информации

Защита информации — комплекс мероприятий, направленных на обеспечение важнейших аспектов информационной безопасности (целостности, доступности и, если нужно, конфиденциальности информации и ресурсов, используемых для ввода, хранения, обработки и передачи данных).

Система называется безопасной, если она, используя соответствующие аппаратные и программные средства, управляет доступом к информации так, что только должным образом авторизованные лица или же действующие от их имени процессы получают право читать, писать, создавать и удалять информацию.

Очевидно, что абсолютно безопасных систем нет, и здесь речь идет о надежной системе в смысле «система, которой можно доверять» (как можно доверять человеку). Система считается надежной, если она с использованием достаточных аппаратных и программных средств обеспечивает одновременную обработку информации разной степени секретности группой пользователей без нарушения прав доступа.

Основными критериями оценки надежности являются: политика безопасности и гарантированность.

Политика безопасности, являясь активным компонентом защиты (включает в себя анализ возможных угроз и выбор соответствующих мер противодействия), отображает тот набор законов, правил и норм поведения, которым пользуется конкретная организация при обработке, защите и распространении информации.

Выбор конкретных механизмов обеспечения безопасности системы производится в соответствии со сформулированной политикой безопасности.

Гарантированность, являясь пассивным элементом защиты, отображает меру доверия, которое может быть оказано архитектуре и реализации системы (другими словами, показывает, насколько корректно выбраны механизмы, обеспечивающие безопасность системы).

В надежной системе должны регистрироваться все происходящие события, касающиеся безопасности (должен использоваться механизм подотчетности протоколирования, дополняющийся анализом запомненной информации, то есть аудитом).

При оценке степени гарантированное, с которой систему можно считать надежной, центральное место занимает достоверная (надежная) вычислительная база. Достоверная вычислительная база (ДВБ) представляет собой полную совокупность защитных механизмов компьютерной системы, которая используется для претворения в жизнь соответствующей политики безопасности.

Надежность ДВБ зависит исключительно от ее реализации и корректности введенных данных (например, данных о благонадежности пользователей, определяемых администрацией).

Граница ДВБ образует периметр безопасности. Компоненты ДВБ, находящиеся внутри этой границы, должны быть надежными (следовательно, для оценки надежности компьютерной системы достаточно рассмотреть только ее ДВБ). От компонентов, находящихся вне периметра безопасности, вообще говоря, не требуется надежности. Однако это не должно влиять на безопасность системы. Так как сейчас широко применяются распределенные системы обработки данных, то под «периметром безопасности» понимается граница владений определенной организации, в подчинении которой находится эта система. Тогда по аналогии то, что находится внутри этой границы, считается надежным. Посредством шлюзовой системы, которая способна противостоять потенциально ненадежному, а может быть даже и враждебному окружению, осуществляется связь через эту границу.

Контроль допустимости выполнения субъектами определенных операций над объектами, то есть функции мониторинга, выполняется достоверной вычислительной базой. При каждом обращении пользователя к программам или данным монитор проверяет допустимость данного обращения (согласованность действия конкретного пользователя со списком разрешенных для него действий). Реализация монитора обращений называется ядром безопасности, на базе которой строятся все защитные механизмы системы. Ядро безопасности должно гарантировать собственную неизменность.

Защита ПК от несанкционированного доступа

Как показывает практика, несанкционированный доступ (НСД) представляет

одну из наиболее серьезных угроз для злоумышленного завладения защищаемой информацией в современных АСОД. Как ни покажется странным, но для ПК опасность данной угрозы по сравнению с большими ЭВМ повышается, чему способствуют следующие объективно существующие обстоятельства:
1) подавляющая часть ПК располагается непосредственно в рабочих комнатах специалистов, что создает благоприятные условия для доступа к ним посторонних лиц;
2) многие ПК служат коллективным средством обработки информации, что обезличивает ответственность, в том числе и за защиту информации;
3) современные ПК оснащены несъемными накопителями на ЖД очень большой емкости, причем информация на них сохраняется даже в обесточенном состоянии;
4) накопители на ГД производятся в таком массовом количестве, что уже используются для распространения информации так же, как и бумажные носители;
5) первоначально ПК создавались именно как персональное средство автоматизации обработки информации, а потому и не оснащались специально средствами защиты от НСД.

В силу сказанного те пользователи, которые желают сохранить конфиденциальность своей информации, должны особенно позаботиться об оснащении используемой ПК высокоэффективными средствами защиты от НСД.

Основные механизмы защиты ПК от НСД могут быть представлены следующим перечнем:
1) физическая защита ПК и носителей информации;
2) опознавание (аутентификация) пользователей и используемых компонентов обработки информации;
3) разграничение доступа к элементам защищаемой информации;
4) криптографическое закрытие защищаемой информации, хранимой на носителях (архивация данных);
5) криптографическое закрытие защищаемой информации в процессе непосредственной ее обработки;
6) регистрация всех обращений к защищаемой информации.
Ниже излагаются общее содержание и способы использования перечисленных механизмов.

Теперь о безопасности информации в базах данных. В современных СУБД поддерживается один из двух наиболее общих подходов к вопросу обеспечения безопасности данных: избирательный подход и обязательный подход. В обоих подходах единицей данных или «объектом данных», для которых должна быть создана система безопасности, может быть как вся база данных целиком, так и

любой объект внутри базы данных.

Эти два подхода отличаются следующими свойствами:

В случае избирательного управления некоторый пользователь обладает различными правами (привилегиями или полномочиями) при работе с данными объектами. Разные пользователи могут обладать разными правами доступа к одному и тому же объекту. Избирательные права характеризуются значительной гибкостью.
В случае избирательного управления, наоборот, каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска. При таком подходе доступом к определенному объекту данных обладают только пользователи с соответствующим уровнем допуска.

Для реализации избирательного принципа предусмотрены следующие методы.
В базу данных вводится новый тип объектов БД — это пользователи.
Каждому пользователю в БД присваивается уникальный идентификатор.
Для дополнительной защиты каждый пользователь кроме уникального идентификатора снабжается уникальным паролем, причем если идентификаторы пользователей в системе доступны системному администратору, то пароли пользователей хранятся чаще всего в специальном кодированном виде и известны только самим пользователям.

Пользователи могут быть объединены в специальные группы пользователей. Один пользователь может входить в несколько групп. В стандарте вводится понятие группы PUBLIC, для которой должен быть
определен минимальный стандартный набор прав. По умолчанию предполагается, что каждый вновь создаваемый пользователь, если специально не указано иное, относится к группе PUBLIC.

Привилегии или полномочия пользователей или групп — это набор действий (операций), которые они могут выполнять над объектами БД.

В последних версиях ряда коммерческих СУБД появилось понятие «роли». Роль — это поименованный набор полномочий. Существует ряд стандартных ролей, которые определены в момент установки сервера баз данных.
И имеется возможность создавать новые роли, группируя в них произвольные полномочия. Введение ролей позволяет упростить управление привилегиями пользователей, структурировать этот процесс. Кроме того, введение ролей не связано с конкретными пользователями, поэтому роли могут быть определены и сконфигурированы до того, как определены пользователи системы.

Пользователю может быть назначена одна или несколько ролей.

Объектами БД, которые подлежат защите, являются все объекты, хранимые в БД: таблицы, представления, хранимые процедуры и триггеры. Для каждого типа объектов есть свои действия, поэтому для каждого типа объектов могут быть определены разные права доступа.

На самом элементарном уровне концепции обеспечения безопасности баз данных исключительно просты. Необходимо поддерживать два фундаментальных принципа: проверку полномочий и проверку подлинности (аутентификацию).

Проверка полномочий основана на том, что каждому пользователю или процессу информационной системы соответствует набор действий, которые он может выполнять по отношению к определенным объектам. Проверка подлинности означает достоверное подтверждение того, что пользователь или процесс, пытающийся выполнить санкционированное действие, действительно тот, за кого он себя выдает.

Система назначения полномочий имеет в некотором роде иерархический характер. Самыми высокими правами и полномочиями обладает системный администратор или администратор сервера БД. Традиционно только этот тип пользователей может создавать других пользователей и наделять их определенными полномочиями.

СУБД в своих системных каталогах хранит как описание самих пользователей, так и описание их привилегий по отношению ко всем объектам.

Далее схема предоставления полномочий строится по следующему принципу. Каждый объект в БД имеет владельца — пользователя, который создал данный объект. Владелец объекта обладает всеми правами-полномочиями на данный объект, в том числе он имеет право предоставлять другим пользователям полномочия по работе с данным объектом или забирать у пользователей ранее предоставленные полномочия.

В ряде СУБД вводится следующий уровень иерархии пользователей — это администратор БД. В этих СУБД один сервер может управлять множеством СУБД (например, MS SQL Server, Sybase). В СУБД Oracle применяется однобазовая архитектура, поэтому там вводится понятие подсхемы — части общей схемы БД и вводится пользователь, имеющий доступ к подсхеме. В стандарте SQL не определена команда создания пользователя, но практически во всех коммерческих СУБД создать пользователя можно не только в интерактивном режиме, но и программно с использованием специальных хранимых процедур. Однако для выполнения этой операции пользователь должен иметь право на запуск

соответствующей системной процедуры.

В стандарте SQL определены два оператора: GRANT и REVOKE соответственно предоставления и отмены привилегий.

Оператор предоставления привилегий имеет следующий формат:

GRANT {<список действий | ALL PRIVILEGES }
ON <имя_объекта> TO (<имя_пользователя>] PUBLIC } [WITH GRANT OPTION]

Здесь список действий определяет набор действий из общедопустимого перечня действий над объектом данного типа.

Параметр ALL PRIVILEGES указывает, что разрешены все действия из допустимых для объектов данного типа.

<имя_обьекта> — задает имя конкретного объекта:
таблицы, представления, хранимой процедуры, триггера.
<имя_пользователя> или PUBLIC определяет, кому предоставляются данные привилегии.

Параметр WITH GRANT OPTION является необязательным и определяет режим, при котором передаются не только права на указанные действия, но и право передавать эти права другим пользователям. Передавать права в этом случае пользователь может только в рамках разрешенных ему действий.

Рассмотрим пример, пусть у нас существуют три пользователя с абсолютно уникальными именами userl, user2 и user3. Все они являются пользователями одной БД.

User1 создал объект Tab1, он является владельцем этого объекта и может передать права на работу с эти объектом другим пользователям. Допустим, что пользователь user2 является оператором, который должен вводить данные в Tab1 (например, таблицу новых заказов), а пользователь user 3 является большим начальником (например, менеджером отдела), который должен регулярно просматривать введенные данные.

Для объекта типа таблица полным допустимым перечнем действий является набор из четырех операций: SELECT, INSERT, DELETE, UPDATE. При этом операция обновление может быть ограничена несколькими столбцами.

Общий формат оператора назначения привилегий для объекта типа таблица будет иметь следующий синтаксис:

GRANT {[SELECT][.INSERT][,DELETED[.UPDATE (<список столбцов>)]} ON <имя таблицы>
TO {<имя_пользователя> PUBLIC }
[WITH GRANT OPTION]

Тогда резонно будет выполнить следующие назначения:

GRANT INSERT
ON Tab1
TO user2 GRANT SELECT
ON Tab1
TO user3

Эти назначения означают, что пользователь user2 имеет право только вводить новые строки в отношение Tab1> а пользователь user3 имеет право просматривать все строки в таблице Tab1.

При назначении прав доступа на операцию модификации можно уточнить, значение каких столбцов может изменять пользователь. Допустим, что менеджер отдела имеет право изменять цену на предоставляемые услуги. Предположим, что цена задается в столбце COST таблицы Tab1. Тогда операция назначения привилегий пользователю user3 может измениться и выглядеть следующим образом:

GRANT SELECT. UPDATE (COST) ON Tab1 TO user3

Если наш пользователь user1 предполагает, что пользователь user4 может его замещать в случае его отсутствия, то он может предоставить этому пользователю все права по работе с созданной таблицей Tab1.

GRANT ALL PRIVILEGES
ON Tab1
TO user4 WITH GRANT OPTION

В этом случае пользователь user4 может сам назначать привилегии по работе с таблицей Tab1 в отсутствие владельца объекта пользователя user1. Поэтому в случае появления нового оператора пользователя user5 он может назначить ему права на ввод новых строк в таблицу командой

GRANT INSERT
ON Tab1 TO user5

Если при передаче полномочий набор операций над объектом ограничен, то пользователь, которому переданы эти полномочия, может передать другому пользователю только те полномочия, которые есть у него, или часть этих полномочий. Поэтому если пользователю user4 были делегированы следующие полномочия:

GRANT SELECT. UPDATE. DELETE
ON Tab1
TO user4 WITH GRANT OPTION,

то пользователь user4 не сможет передать полномочия на ввод данных пользователю user5, потому что эта операция не входит в список разрешенных для

него самого.

Кроме непосредственного назначения прав по работе с таблицами эффективным методом защиты данных может быть создание представлений, которые будут содержать только необходимые столбцы для работы конкретного пользователя и предоставление прав на работу с данным представлением пользователю.

Так как представления могут соответствовать итоговым запросам, то для этих представлений недопустимы операции изменения, и, следовательно, для таких представлений набор допустимых действий ограничивается операцией SELECT. Если же представления соответствуют выборке из базовой таблицы, то для такого представления допустимыми будут все 4 операции: SELECT, INSERT, UPDATE и DELETE.

Для отмены ранее назначенных привилегий в стандарте SQL определен оператор REVOKE. Оператор отмены привилегий имеет следующий синтаксис:
REVOKE {<список операций | ALL PRIVILEGES} ON <имя_объекта>
FROM {<список пользователей | PUBLIC } {CASCADE | RESTRICT }

Параметры CASCADE или RESTRICT определяют, каким образом должна производиться отмена привилегий. Параметр CASCADE отменяет привилегии не только пользователя, который непосредственно упоминался в операторе GRANT при предоставлении ему привилегий, но и всем пользователям, которым этот пользователь присвоил привилегии, воспользовавшись параметром WITH GRANT OPTION.

Например, при использовании операции:
REVOKE ALL PRIVILEGES - ON Tab1 TO user4 CASCADE
будут отменены привилегии и пользователя user5, которому пользователь user4 успел присвоить привилегии.

Параметр RESTRICKT ограничивает отмену привилегий только пользователю, непосредственно упомянутому в операторе REVOKE. Но при наличии делегированных привилегий этот оператор не будет выполнен. Так, например, операция:
REVOKE ALL PRIVILEGES ON Tab1 TO user4 RESTRICT
не будет выполнена, потому что пользователь user4 передал часть своих полномочий пользователю user5.

Посредством оператора REVOKE можно отобрать все или только некоторые из ранее присвоенных привилегий по работе с конкретным объектом. При этом из описания синтаксиса оператора отмены привилегий видно, что можно отобрать привилегии одним оператором сразу у нескольких пользователей или у целой

группы PUBLIC.

Поэтому корректным будет следующее использование оператора REVOKE:

REVOKE INSERT ON Tab! TO user2.user4 CASCADE

При работе с другими объектами изменяется список операций, которые используются в операторах GRANT и REVOKE.

По умолчанию действие, соответствующее запуску (исполнению) хранимой процедуры, назначается всем членам группы PUBLIC.

Если вы хотите изменить это условие, то после создания хранимой процедуры необходимо записать оператор REVOKE.

REVOKE EXECUTE ON COUNT_EX TO PUBLIC CASCADE

И теперь мы можем назначить новые права пользователю user4.

GRANT EXECUTE ON COUNT_EX TO user4

Системный администратор может разрешить некоторому пользователю создавать и изменять таблицы в некоторой БД. Тогда он может записать оператор предоставления прав следующим образом:

GRANT CREATE TABLE. ALTER TABLE, DROP TABLE ON OB_LIB TO user1

В этом случае пользователь user1 может создавать, изменять или удалять таблицы в БД DB_LIB, однако он не может разрешить создавать или изменять таблицы в этой БД другим пользователям, потому что ему дано разрешение без права делегирования своих возможностей.

В некоторых СУБД пользователь может получить права создавать БД. Например, в MS SQL Server системный администратор может предоставить пользователю main_user право на создание своей БД на данном сервере. Это может быть сделано следующей командой:

GRANT CREATE DATABASE
ON SERVERJ) TO main user

По принципу иерархии пользователь main_user, создав свою БД, теперь может предоставить права на создание или изменение любых объектов в этой БД другим пользователям. В СУБД, которые поддерживают однобазовую архитектуру, такие разрешения недопустимы. Например, в СУБД Oracle на сервере создается только одна БД, но пользователи могут работать на уровне подсхемы (части таблиц БД и связанных с ними объектов). Поэтому там вводится понятие системных привилегий. Их очень много, 80 различных привилегий.

Для того чтобы разрешить пользователю создавать объекты внутри этой БД, используется понятие системной привилегии, которая может быть назначена одному или нескольким пользователям. Они выдаются только на действия и конкретный тип объекта. Поэтому- если вы, как системный

администратор, предоставили пользователю право создания таблиц (CREATE TABLE), то для того чтобы он мог создать триггер для таблицы, ему необходимо предоставить еще одну системную привилегию CREATE TRIGGER. Система защиты в Oracle считается одной из самых мощных, но это имеет и обратную сторону — она весьма сложная. Поэтому задача администрирования в Oracle требует хорошего знания как семантики принципов поддержки прав доступа, так и физической реализации этих возможностей.

2 Реализация защиты в некоторых СУБД

2.1 Архитектура защиты Access

Если у вас имеется опыт работы с защитой, используемой на сервере или большой ЭВМ, структура защиты в Access покажется вам знакомой. Вы можете указать пользователей, которым предоставляется или, наоборот, не разрешается доступ к объектам базы данных. Кроме того, вы можете определить группы пользователей и назначить разрешения на уровне группы, чтобы облегчить построение защиты для большого числа пользователей. Пользователю достаточно быть членом группы, чтобы получить права доступа, установленные для неё.

Access хранит информацию о защите в двух местах. Во время установки программа Setup создаст в папке \Program Files\Microsoft Ofice\0ffice стандартный файл рабочей группы (System.mdw), который впоследствии используется по умолчанию при запуске Access. Этот файл содержит информацию обо всех пользователях и группах. При создании базы данных Access сохраняет сведения о правах, предоставляемых конкретным пользователям и группам, в файле базы данных.

Общая структура защиты Access отображена на рисунке 1 (см. Приложение А). Учётные записи пользователей и групп хранятся в файле рабочей группы. Разрешение на доступ к конкретным объектам сохраняются в файле базы данных.

Расположение текущего файла рабочей группы хранится в реестре Windows. Можно использовать служебную программу Wrkadm.exe (администратор рабочих групп) для изменения текущего или определения нового файла рабочей группы. Кроме того, можно выбирать нужный файл рабочей группы во время выполнения приложения, задав соответствующий параметр командной строки в ярлыке запуска. Если вам приходится часто запускать в сети совместно используемое защищенное приложение, нужно позаботиться о том, чтобы системный администратор задал вашу рабочую группу, используемую по умолчанию, как общий файл в сетевой папке.

Каждая рабочая группа имеет уникальный внутренний идентификатор, генерируемый Access при определении файла рабочих групп. Любая база данных, созданная пользователем рабочей группы, «принадлежит» как этому пользователю, так и рабочей группе. Каждый пользователь и группа также имеет уникальный внутренний идентификатор, но можно дублировать один и тот же код пользователя и группы в нескольких рабочих группах. Когда вы назначаете право доступа к объекту своей базы данных, Access сохраняет в ней внутренний идентификатор пользователя или группы вместе с информацией о доступе. Таким образом, предоставленные вами права перемещаются вместе с файлом базы данных при копировании его в другую папку или на другой компьютер.

2.2 MS SQL Server: Защита, управление доступом

2.2.1 Защита и доступ

В критических для бизнеса приложениях, когда сервер СУБД должен быть постоянно доступен для клиентов, большинство профилактических работ по поддержке базы данных приходится выполнять фактически в режиме on - line. MS SQL Server обладает возможностями динамического резервного копирования данных, т. е. даже когда эти данные используются и изменяются клиентами. В

случае сбоя оборудования, отключения питания и т. д. механизм автоматического восстановления MS SQL Server восстанавливает все базы данных до их последнего целостного состояния без вмешательства администратора. Все завершенные, но не отраженные в базе транзакции из журнала транзакций применяются к базе данных (это фактически то, что происходит при событии chekpoint), а незавершенные транзакции, т. е. те, которые были активными на момент сбоя, вычищаются из журнала.

Говоря о симметричной архитектуре, операции резервного копирования и восстановления могут распараллеливаться на несколько потоков и выполняться одновременно, используя преимущества асинхронного ввода/вывода. На каждое резервное устройство отводится свой поток. Параллельное резервное копирование поддерживает до 32 одновременных резервных устройств (backup devices), что позволяет быстро создавать страховочные копии баз данных даже очень большой емкости. Возможность резервного копирования и восстановления отдельных таблиц, о чем мы упоминали, рассматривая Transact-SQL, позволяет экономить место и время, не выполняя копирование всей базы ради только некоторых ее объектов. Однако резервное копирование отдельной таблицы требует наложения на нее блокировки exclusive в отличие от резервного копирования всей базы или журнала транзакций, которые могут выполняться независимо от степени активности пользователей. Резервным копиям может быть назначен предельный срок хранения или дата утраты актуальности, до наступления которой место, занятое на устройстве этими копиями, не может использоваться для размещения других резервных копий при инициализации устройства. В качестве резервных устройств могут также применяться временные устройства, не входящие в состав базы и не имеющие записей в системной таблице sysdevices:

```
DECLARE @tomorrow char(8)
SELECT @tomorrow = CONVERT(char(8), DATEADD(dd, 1, GETDATE()) , 1)
DUMP DATABASE pubs
TO DISK = '\\ntalexeysh\disk_d\sql_experiments\pubs.dmp'
WITH INIT, EXPIREDATE=@tomorrow, STATS
```

Для небольшой базы данных ее журнал транзакций обычно хранится на том же устройстве, что и сама база, и архивируется вместе с ней. Журналирование транзакций ведется по принципу write-ahead, что означает, что любое изменение сначала отражается в журнале транзакций и лишь потом попадает собственно в базу. В случае нахождения журнала транзакций на отдельном устройстве существует возможность отдельного резервного копирования журнала транзакций.

Как правило, резервное копирование базы данных организуется с меньшей частотой, чем журнала транзакций. Например, сохранение журнала транзакций выполняется ежедневно, а страховая копия всей базы может делаться раз в неделю, так как архивирование журнала транзакций происходит значительно быстрее по времени и занимает меньше места, чем дамп целой базы. В отличие от резервирования базы данных дамп журнала транзакций очищает его неактивную часть, т. е. все завершившиеся (зафиксированные или абортированные) с момента последнего дампа транзакции, если только не использована опция NO_TRUNCATE. Команда DUMP TRANSACTION TRUNCATE_ONLY, очищающая журнал транзакций, полезна в случае его переполнения, которое можно контролировать, например, оператором DBCC SQLPERF (LOGSPACE). Если степень переполнения журнала очень высока, можно при его очистке отказаться от журналирования факта самого этого события: DUMP TRANSACTION NO_LOG. Если резервное копирование транзакций не представляет интереса, можно включить опцию очистки последних завершенных транзакций в базе по наступлению события checkpoint. Смысл механизма checkpoint состоит в периодической записи данных из кэша на диск, чтобы не допускать грязных данных. Такого рода события постоянно генерируются MS SQL Server или возникают по инициативе пользователя. Включенная опция truncate log on checkpoint гарантирует выполнение с определенной частотой обработчиком события действий, приблизительно эквивалентных команде DUMP TRANSACTION TRUNCATE_ONLY.

Заключение

При восстановлении журнала транзакций соответствующие транзакции применяются к базе данных. Это означает, что если в начале недели была сделана резервная копия всей базы, а потом ежедневно архивировались транзакции за каждый день, то при необходимости восстановления поднимается состояние базы на начало недели и на него последовательно накатываются дампы журнала транзакций за все дни, предшествующие моменту восстановления. MS SQL Server 6.5 имеет возможность восстановления данных из журнала транзакций на произвольный момент времени (разумеется, отраженный в журнале) при помощи команды LOAD TRANSACTION STOPAT <t> или в окне database backup and restore выбором опции until time. Все содержащиеся в этом дампе транзакции, отмеченные завершившимися после этого момента, будут откачены.

Источники литературы

https://support.microsoft.com/ru-ru/office/безопасность-в-access-2010-cae6d764-0318-4622-955f-68d9f186d6ca
https://poznayka.org/s2707t2.html