В. Н. Федосеев
ФГУП «НИИ НПО “Луч”», (г. Подольск)

МНОГОУРОВНЕВЫЕ ОБЪЕКТЫ УПРАВЛЕНИЯ ДОСТУПОМ

Проведен анализ многоуровневых субъектов и объектов мандатного управления доступом, отличающийся от традиционной модели Белла-ЛаПадулы. Альтернативой использованию сертифицированных многоуровневых СУБД является декомпозиция многоуровневой базы данных на несколько одноуровневых и применение сертифицированных одноуровневых СУБД.

Введение

Многоуровневая безопасность MLS обычно определяется как способность компьютерной системы обрабатывать данные с различной чувствительностью и поддерживать одновременный санкционированный доступ пользователей с различным допуском. Чувствительность данных может трактоваться различным образом, в том числе так, как это определено законом РФ «О государственной тайне». В предыдущей статье [1] было показано, что, организуя определенным образом многоуровневые данные на уровне файлов и используя строгое изолирование ресурсов компьютера в виртуальных машинах с контролем потоков информации между ними, можно реализовать мандатное управление доступом по модели Белла-ЛаПадулы BLP [2] на недоверенных операционных системах. Указанная организация заключалась в преобразовании (декомпозиции) многоуровневых MLS-систем в набор одноуровневых MSL-систем.

Однако на практике нередко приходится сталкиваться с многоуровневыми объектами, чьи различные части являются неоднородно классифицированными. Примеры таких объектов — это диски накопителей информации, файловые каталоги, базы данных. Для обработки подобных объектов могут использоваться доверенные многоуровневые субъекты, например СУБД. Проблема обработки многоуровневых объектов достаточно актуальна и заслуживает более подробного рассмотрения.

1. Многоуровневые объекты, субъекты и операции

Классическая модель Белла-ЛаПадулы неприменима к сложным составным объектам. Для разрешения этого противоречия Белл [3] предложил вместо текущей метки субъекта s применять максимальную метку для чтения $sl_{r\text{-}max}(s)$ и минимальную метку для записи $sl_{w\,min}(s)$. Но дальше этого дело не пошло. Современное расширение модели Белла-ЛаПадулы описывает состояние безопасности сложного составного объекта следующей четверкой параметров [4]:

Q=<b, M, f, H>,

где b=<s, o, m> — текущее состояние доступа, s — субъект, o — объект, m — режим доступа; M — матрица дискреционного доступа M[s, o]; f — функция, определяющая метку мандатного управления доступом на каждом объекте и субъекте: $f : O \bigcup S \rightarrow SL$; H — функция, описывающая иерархию объектов, $H : O \rightarrow P(O)$, P — разрешения на различных наборах объектов. В функции f выделяют три составляющие: допуск пользователя $sl_{max}(s)$, текущую метку пользователя $sl_{cur}(s)$, метку объекта sl(o). Метка каждого объекта в иерархии должна доминировать над меткой его родителей. Метка пользователя может меняться в течение сессии при условии: $sl_{max}(s)$ dom $sl_{cur}(s)$, где dom обозначает операцию доминирования.

Существование многоуровневых объектов неизбежно влечет появление многоуровневых субъектов, которые могут проводить операции на одном или нескольких уровнях. Примером является СУБД, база данных и транзакции с объектами баз данных. В качестве дополнительных примеров можно привести составные документы и многоуровневые каталоги (директории), в которых содержатся файлы различной чувствительности.

Пусть $sl_{max}(o)$, $sl_{min}(o)$ — максимальная и минимальная метки объекта o. Разница между ними $d(o)= sl_{max}(o) - sl_{min}(o)$ характеризует неоднородность классификации многоуровневого объекта. Для

модели BLP $d(o) = 0 \quad \forall o \in O$. Разницу $d(s)=sl_{r\text{-}max}(s) - sl_{w\,min}(s)$ можно рассматривать как величину доверия к субъекту. Для недоверенного субъекта $d(s)=0$ (как в модели BLP). Для $d(s)>0$ субъект должен быть доверенным и пройти соответствующую верификацию. Доверие здесь понимается в общепринятом смысле как ожидание того, что некоторая программа ведет себя положенным образом для специфической цели. Будем считать, что субъект s осуществляет многоуровневую операцию над объектом o, если $d(s) \neq 0 \wedge d(o) \neq 0$, причем действие операции проявляется на всех уровнях объекта. В качестве примера многоуровневой операции можно привести удаление всех файлов из многоуровневого каталога. Тогда два фундаментальных свойства модели BLP для многоуровневых субъектов и объектов можно записать следующим образом:

	Операции	
	Одноуровневые	Многоуровневые
Чтение	$sl_{r\text{-}max}(s)$ dom $sl_{min}(o)$	$sl_{r\text{-}max}(s)$ dom $sl_{max}(o)$
Запись	$sl_{max}(o)$ dom $sl_{w\text{-}min}(s)$	$sl_{min}(o)$ dom $sl_{w\text{-}min}(s)$

Если доверенный субъект имеет право на операцию чтения или записи, то должны соблюдаться соотношения доминирования, указанные в таблице (но не наоборот). Объектная иерархия с доминированием текущего объекта над родительским применительно к многоуровневым файловым каталогам трансформируется следующим образом: метки чувствительности на них должны монотонно не убывать, начиная от корневого каталога. В противном случае часть каталогов может оказаться невидимой для некоторых пользователей.

Приведенные в таблице соотношения устанавливают требования для многоуровневых субъектов, объектов и операций. Однако верификация и сертификация СУБД представляет собой сложный и трудоемкий процесс. В качестве известных примеров можно назвать Trusted Oracle или российский Линтер [5]. Было бы желательно перенести часть доверия с СУБД на операционную систему, при условии, что последняя поддерживает метки $sl_{r\text{-}max}(s)$ и $sl_{w\text{-}min}(s)$.

Проект RSBAC (www.rsbac.org) реализует концепцию динамических меток, когда текущая метка субъекта может повышаться или понижаться в пределах установленного допуска. Оба фундаментальных свойства безопасности при этом не нарушаются. Допустим, пользователь хочет прочитать файл с более высокой меткой, чем текущая метка процесса, но ниже допуска этого пользователя. Текущая метка повышается, позволяя пользователю читать информацию на более высокой метке. Одновременно пользователю запрещается писать на прежней, более низкой метке. Для отслеживания процесса динамического изменения меток в RSBAC имеется структура, описывающая процесс с шестью метками безопасности, включая допуск, текущую метку, минимальную метку для записи и максимальную метку для чтения. Терминологически это означает, что RSBAC соблюдает свойство слабой статичности (weak tranquility).

2. Многоуровневые СУБД

Концепция многоуровневой безопасности в равной степени применима и к системам управления базами данных. Однако обеспечение безопасности на уровне ОС и СУБД существенно различается. Специфика проявляется в субъектах доступа (транзакция вместо процесса), объектах (базы данных, таблицы, записи и пр. вместо файлов и каталогов), операциях (например, INSERT, UPDATE, CREATE, DELETE вместо write), в наличии ограничений, многоуровневых транзакций, семантических корреляций, метаданных, логических представлений. Модель Sea View [6] обычно рассматривается как основополагающая в данной области. Различают детальность объектов для маркировки метками мандатного доступа или гранулярность: на уровне реляции, кортежа, атрибута или элемента данных. Схема многоуровневой реляции записывается следующим образом [7]:

$MREL(A_1, C_1, A_2, C_2, \ldots A_n, C_n, TC)$,

где A_i — i-й атрибут реляции с классификацией C_i; $A_i \in D_i$; D_i — домен, в котором определен тип атрибута A_i; домен каждой классификации C_i определяется как множество {Li…Hi} ($H_i \geq L_i$); i=1, 2,… n; TC — классификация кортежа: $TC = \bigcup_{i=1}^{n}(\{L_i ... H_i\})$.

Назначение меток на уровне реляций является негибким, а установка меток на уровне кортежей или элементов может приводить к таким проблемам, как существование нескольких экземпляров данных (polyinstantiation).

Основные архитектуры многоуровневых СУБД были определены на конференции в Woods Hole [8], где были предложены четыре основные архитектуры:

· *Интегрированная*. Примерами коммерческих продуктов являются Sybase, Rubix, Informix, Oracle, Линтер. Иногда называется архитектурой с доверенным субъектом. Фактически это полноценная реализация многоуровневой СУБД.

· *Блокировки целостности*. Примером является продукт Trudata. Целостность хранимых данных обеспечивается криптографическими методами, а пользовательский интерфейс содержит доверенный фильтр, осуществляющий шифрацию и дешифрацию данных и меток.

· *Фрагментированная*. Примером коммерческого продукта является Oracle с маркировкой кортежей, а примером исследовательского проекта — Sea View с маркировкой элементов. Проводилась декомпозиция на несколько баз данных с одним уровнем чувствительности каждая. С каждой меткой данных работал отдельный экземпляр СУБД.

· *Репликационная*. Коммерческих продуктов нет, но имелся исследовательский проект SINTRA. Также проводилась декомпозиция данных на одноуровневые базы данных. Но в отличие от фрагментированной архитектуры i-я база данных содержала данные i-го уровня чувствительности по чтению-записи и все данные более низких уровней по чтению. Другими словами, декомпозиция осуществлялась не на одноуровневые данные, а по принципу наивысшей метки. Такая архитектура требует доверенного механизма для репликации данных на более высокие уровни.

Интегрированная архитектура требует значительных усилий при разработке и сертификации. Естественно, что многоуровневая СУБД значительно сложнее в эксплуатации. База данных должна проектироваться с учетом ряда ограничений. Требования ссылочной целостности в этом случае трансформируются следующим образом [9]: 1) каждый внешний ключ и главный ключ должны быть однородно классифицированы, т. е. все атрибуты в составе этих ключей должны иметь одну и ту же метку безопасности; 2) каждый внешний ключ должен доминировать над соответствующим главным ключом. Клиенты многоуровневых СУБД, как правило, являются одноуровневыми, что привносит существенные сложности в разработку приложений.

Фрагментированная и репликационная архитектуры допускают применение коммерческих СУБД, не имеющих в своем составе мандатного управления доступом. Ранее фрагментированная архитектура предлагалась автором в качестве перспективного подхода для российских систем учета и контроля ядерных материалов [10]. Однако проведенный дополнительный анализ показал сложность создания достаточно производительного репликационного механизма. С позиций сегодняшнего дня фрагментированная архитектура выглядит предпочтительнее.

3. Декомпозиция и реконструкция БД

В общем случае разработка многоуровневых баз данных и приложений для них должна пройти следующие основные этапы:

· проектирование структуры многоуровневой базы данных;

· дефрагментация многоуровневой базы данных на несколько одноуровневых. Как правило, число одноуровневых баз данных составляет две-три, а число реляций в каждой из них убывает с ростом чувствительности;

· обеспечение ссылочной целостности в рамках каждой одноуровневой базы данных и противодействие появлению нескольких экземпляров данных (polyinstantiation);

· создание и инициализация баз данных. Каждой одноуровневой базе данных средствами операционной системы назначаются соответствующие метки мандатного доступа;

· разработка приложений (по одному на каждом уровне). Каждое приложение пишет данные на своем уровне и читает данные на более низких уровнях. Поскольку СУБД контролирует ссылочную целостность лишь в пределах отдельных одноуровневых баз данных, согласование последних между собой должно взять на себя приложение;

· назначение меток приложениям каждого уровня;

· развертывание приложений. Создание учетных записей и назначение допусков пользователям;

· эксплуатация приложений и баз данных.

В достаточно общем виде декомпозиция рассмотрена в [11]. Вместе с тем во многих практических случаях многоуровневые базы данных имеют относительно простую структуру или объем чувствительных данных составляет небольшую долю. Кроме того, хранимая информация часто образует лишь одну категорию. В таких вырожденных случаях разработчики могут свести одну многоуровневую базу данных к нескольким одноуровневым, не прибегая к сложным преобразованиям. Пример практического подхода к декомпозиции предложен российскими исследователями в [12]. Не вдаваясь в детали, ниже будут рассмотрены два частных, но достаточно характерных случая проектирования базы данных с горизонтальной и вертикальной фрагментацией.

1. Маркировка на уровне кортежей

Будем считать, что в многоуровневой базе данных содержится достаточно много реляций, но лишь одна или несколько из них содержат чувствительную информацию. Предполагается, что вся информация в кортеже однородно классифицирована, а набор всех кортежей описывается множеством: $(a_{1k}, a_{2k},\ldots a_{nk}, t_{ck})$.

После этого каждую чувствительную реляцию следует разбить на несколько, добиваясь, чтобы каждая новая реляция была одноуровневой. Одноуровневые таблицы следует разместить по базам данных соответствующего уровня.

2. Реляция с единственным чувствительным атрибутом

Когда реляция содержит значительное число атрибутов, ее нельзя рассматривать как классифицированную однородно. При этом метка ставится не на весь кортеж, а лишь на отдельный атрибут. В таких случаях сначала используется вертикальная фрагментация. На первом этапе исходная реляция преобразуется в две, являющиеся ее проекциями. Первая реляция содержит чувствительный атрибут, его классификацию и дополнительный атрибут для связи (но не для ссылочной целостности). Вторая реляция является неклассифицированной и включает все оставшиеся атрибуты, а также дополнительный атрибут для связи с первой реляцией. Далее к первой реляции применяется горизонтальная фрагментация, описанная в первом пункте.

4. Реализация фрагментированной архитектуры

В существующей фрагментированной архитектуре присутствует, по крайней мере, два неясных момента: 1) как обеспечить ссылочную целостность базы данных; 2) как предоставить доступ только по чтению к базам данных с более низкими метками средствами операционной системы.

Проблема ссылочной целостности заключается в том, что после декомпозиции единой многоуровневой базы данных ее семантическая связность нарушается. Базы данных с низкими метками ничего не знают о существовании баз данных с более высокими метками. В противоположном направлении доступ осуществляется лишь по чтению. По мнению автора, возможен компромисс, заключающийся в том, что проблема ссылочной целостности разбивается на две части. За ссылочную целостность одноуровневых баз данных, полученных в результате декомпозиции многоуровневой базы данных,

отвечает СУБД, а задача синхронизации одноуровневых баз данных между собой должна быть возложена на приложения. Конечно, такое решение нельзя рассматривать как полностью надежное. Но оно является вполне приемлемым с точки зрения безопасности, поскольку конфиденциальность данных не нарушается. Ситуация облегчается возможностью периодического сканирования баз данных приложением с наивысшей меткой на предмет выявления потерь целостности. В случае выявления расхождений возможен откат баз данных к требуемому моменту времени или иные меры процедурного характера. Кроме того, современные средства разработки (Java EE, .Net) поддерживают транзакции.

Естественно, что с базой данных в фиксированный момент времени может работать лишь одна СУБД. Она запускается от имени администратора базы данных DBA. С точки зрения операционной системы, доступ к файлам и каталогам, образующим базу данных, также осуществляется администратором, а не пользователями, делающими запросы. Поэтому различные права доступа к базе данных могут осуществляться не на уровне операционной системы, а на прикладном уровне. С учетом двух сделанных уточнений фрагментированная архитектура приобретает следующий вид:

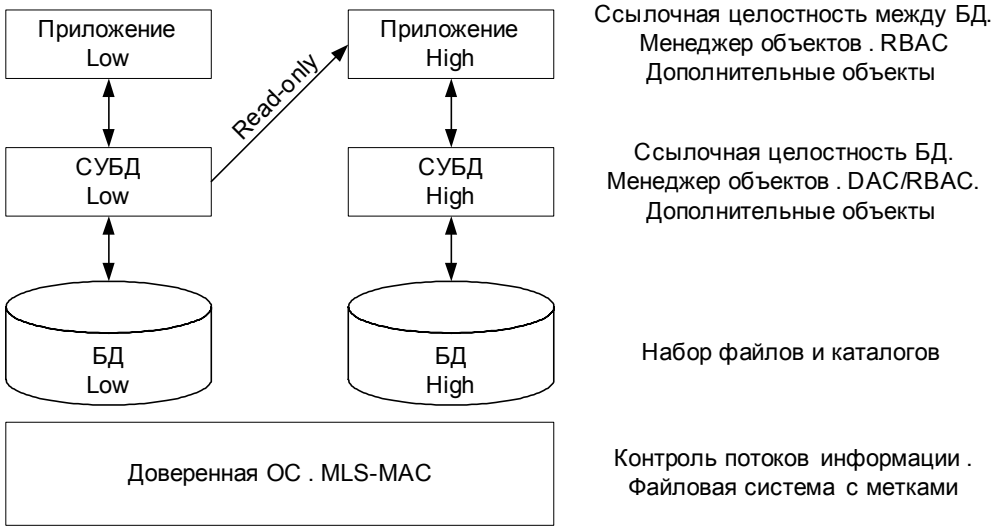

Рис. 1. Реализация фрагментированной архитектуры

В основании архитектуры лежит доверенная операционная система с мандатным управлением доступом. Объектами мандатного доступа являются файлы, каталоги, процессы, межпроцессное взаимодействие IPC и т. д. На файлы и каталоги, образующие одноуровневые базы данных, устанавливаются метки. В приведенном примере имеются две метки — Low и High. DBA запускает экземпляры СУБД, по одному на каждой метке. Приложение взаимодействует по чтению-записи с СУБД на своей метке и со всеми СУБД на более низкой метке по чтению. На приведенном рисунке степень абстракции возрастает снизу вверх. На каждом более высоком уровне появляются новые объекты по отношению к нижележащему уровню.

Возможность доступа приложения с меткой High к базе данных с меткой Low в режиме только для чтения может быть обеспечена средствами дискреционного или ролевого управления доступом СУБД. Например, DBA может выполнить следующую команду:

GRANT SELECT ON TABLE <full_table_list> TO higher_users

Приложение с меткой High должно осуществлять подключение к базе данных с меткой Low от имени пользователей higher_users. Никаких других прав доступа пользователи higher_users не должны иметь.

Для реализации предложенного подхода целесообразно использовать коммерческую СУБД, которая удовлетворяет определенным требованиям безопасности, а именно: наличие аутентификации пользователей, дискреционного управления доступом и аудита. Последняя служба необходима для анализа подключений к базам данных. Естественно, что открытые СУБД (PostgreSQL, Firebird,

Derby…) могут предоставлять дополнительные преимущества, поскольку не требуют лицензионной платы за экземпляры продуктов. Кроме того, их проще сертифицировать в отношении безопасности информации.

Заключение

В работе проведен анализ многоуровневых объектов и субъектов. Установлены ограничения на проведение операций доверенными субъектами. Метки чувствительности на каталогах должны монотонно не убывать, начиная от корневого каталога. Сертификация доверенных субъектов требует серьезных усилий и не всегда возможна. В особой степени это применимо к СУБД. В случае относительно простой структуры БД целесообразно провести ее декомпозицию на несколько одноуровневых баз данных и использовать одноуровневые СУБД.

В данной работе рассмотрена практическая реализация фрагментированной архитектуры, в частности, требования к СУБД, особенности проектирования баз данных и приложений, взаимодействие с доверенной многоуровневой операционной системой. Обеспечение ссылочной целостности в пределах одноуровневых баз данных должно быть возложено на СУБД, а синхронизация баз данных между собой должна осуществляться приложением. Проблемой является ограничение доступа по чтению к базе данных с более низкой меткой. Возможным выходом является перенос этой проблемы с ОС на СУБД. Это означает, что СУБД должна быть сертифицированной, по крайней мере, для механизмов аутентификации, аудита и дискреционного или ролевого управления доступом.

СПИСОК ЛИТЕРАТУРЫ

1. *Федосеев В. Н.* Подход к реализации многоуровневой безопасности в недоверенных операционных системах // Безопасность информационных технологий. 2006. № 3.
2. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. М.: Горячая линия // Телеком. 2000. — 452 с.
3. *Bell D. E.* Security Policy Modeling for the Next-Generation Packet Switch // Proceedings of IEEE Symposium on Security and Privacy. 1988. P. 212—216.
4. *Bishop M.* Computer Security: Art and Science // Addison Wesley Professional. 2003. — 1136 p.
5. *Максимов В. Е.* и др. Защищенная реляционная СУБД ЛИНТЕР // Открытые системы. 1999. № 11—12.
6. *Denning D. et al.* The Sea View Security Model // Proceedings of the IEEE Symposium on Security and Privacy. Washington, DC, 1988. P. 218—233.
7. *Sandhu R., Chen F.* Multilevel Relational (MLR) Data Model // ACM Transactions on Information and System Security, 1998. V. I. №. 1. P. 93—132.
8. *Notargiacomo L.* Architectures for MLS Database Management Systems // Information Security: An Integrated Collection of Essays / M. Abrams, S. Jajodia, H. Podell eds. — IEEE Computer Society. 1995. — 21 p.
9. NCSC Technical Report 005 // Library № S-243,039. USA, National Computer Security Center, 1996. Volume 2/5. — 44 p.
10. *Федосеев В. Н., Мизин П. П., Шанин О. И.* Подход к программному обеспечению российских СУИК следующего поколения // Информационный бюллетень «Новости ФИС». 2003. № 3. Стр. 21—30.
11. *Jajodia S., Sandhu R.* A Novel Decomposition of Multilevel Relation into Single-Level Relations // Proceedings of IEEE Symposium on Security and Privacy. Oakland, CA, USA, 1991. P. 300—313.
12. *Аксарин М. В., Быков Я. А., Тарасюк М. В.* Проектирование приложений реляционных БД с учетом конфиденциальности и целостности баз данных // Безопасность информационных технологий. 2005. № 3. С. 41—49.