D. N. Kolegov, N. O. Tkachenko, D. V. Chernov

The Main Elements of Development of Mandatory Access Control Mechanism in DBMS MySQL based on DP-Models

Keywords: security models, mandatory access control, database management systems
It is proposed an approach to development of mandatory access control in originally discrete DBMS MySQL based on DP-models. It is performed analysis of the existent access control mechanisms in this DBMS, it is described DP-model of MySQL, and it is implemented mandatory access control mechanism based on this model.

Д. Н. Колегов, Н. О. Ткаченко, Д. В. Чернов

ОСНОВНЫЕ ЭЛЕМЕНТЫ РАЗРАБОТКИ МЕХАНИЗМА МАНДАТНОГО УПРАВЛЕНИЯ ДОСТУПОМ В СУБД MYSQL НА ОСНОВЕ ДП-МОДЕЛЕЙ

В настоящее время одной из актуальных задач компьютерной безопасности является разработка и реализация строго обоснованных механизмов мандатного управления доступом и информационными потоками, в том числе в изначально дискреционных системах управления базами данных (СУБД) [1, 2]. Исчерпывающее решение этой задачи предполагает построение и использование адекватной формальной модели безопасности.

В большинстве СУБД используется дискреционное управление доступом. Вместе с тем некоторые изначально дискреционные СУБД (например, Oracle, Microsoft SQL Server, PostgreSQL) имеют механизмы мандатного управления доступом, которые являются наиболее актуальными и востребованными при построении защищенных КС, но при этом обладают следующими основными недостатками:

- не имеют формальной модели политики управления доступом и информационными потоками;
- не учитывают существенное для политики мандатного управления доступом типа MLS требование обеспечения безопасности информационных потоков;
- часто реализуются на прикладном уровне с помощью модификации или перехвата запросов SQL [1, 3, 4].

В то же время, в соответствии с «Критериями оценки безопасности информационных технологий», использование формальных математических моделей является обязательным для КС с высоким уровнем доверия к их безопасности [5]. Стоит отметить, что основные зарубежные подходы к моделированию [6], как правило, не могут быть применены напрямую в отечественных КС [1], что существенно затрудняет применение некоторых подходов (например, [4]) в отечественной практике.

В настоящее время авторами на основе семейства ДП-моделей [1] ведется разработка мандатной ДП-модели для изначально дискреционной СУБД MySQL, учитывающей особенности функционирования последней и адаптированной для реализации в ней механизма мандатного управления доступом. В связи со сложностью этой задачи она решается в несколько этапов. В данной работе проводится анализ существующих механизмов управления доступом в рассматриваемой СУБД, описываются основные элементы текущей мандатной ДП-модели MySQL и на ее основе реализуется прототип механизма мандатного управления доступом.

Анализ механизмов управления доступом в СУБД MySQL

Анализ свойств существующего управления доступом в СУБД MySQL проводился путем изучения документации, исследования исходного кода и выполнения компьютерных экспериментов. На данном этапе исследовались, в основном, информационные потоки по памяти ввиду значительной технической сложности анализа информационных потоков по времени [1].

Механизмы реализации информационных потоков по памяти в СУБД MySQL принципиально отличаются от подобных механизмов в ОС. Например, в современных ОС процесс может иметь доступ на чтение и на запись к нескольким файлам одновременно. В СУБД MySQL субъект-сессия пользователя, как правило, не может иметь доступ на чтение и на запись к двум сущностям СУБД одновременно, исключение составляют механизмы реализации запросов языка SQL следующего вида:

- «INSERT INTO … VALUES((SELECT…), …)»;
- «INSERT … SELECT»;
- «UPDATE … SET … = (SELECT …)».

Подобные запросы языка SQL могут быть использованы для реализации запрещенных информационных потоков по памяти от сущностей с высоким уровнем конфиденциальности к сущностям с низким уровнем конфиденциальности и, тем самым, нарушать требования обеспечения безопасности информационных потоков политики типа MLS. Таким образом, на основе проведенного анализа управления доступом СУБД MySQL были сформулированы следующие предположения, использующиеся далее:

- информационные потоки рассматриваются в пределах СУБД;
- рассматриваются только информационные потоки по памяти, порождаемые операторами SELECT, INSERT, UPDATE и DELETE языка SQL;
- информационные потоки по времени не рассматриваются.

Элементы мандатной ДП-модели СУБД MySQL

Модель строится на основе ДП-моделей общего назначения, мандатных ДП-моделей [1] и СУБД ДП-модели [2]. Модель включает описание элементов как изначальной дискреционной, так и новой мандатной политики управления доступом типа MLS. В настоящей версии ДП-модели MySQL используются следующие обозначения, разработанные на основе [1, 2]:

$O = O_\rho \cup O_t \cup O_v \cup O_{var} \cup O_{gvar} \cup O_c \cup COL$, где O — множество сущностей-объектов, O_ρ — множество сущностей-процедур, O_t — множество сущностей-триггеров, O_v — множество сущностей-представлений, O_{var} — множество сущностей-переменных, O_{gvar} — множество сущностей — глобальных переменных, O_c — множество сущностей-курсоров, COL — множество сущностей-столбцов;

DB — множество контейнеров — баз данных, TAB — множество контейнеров-таблиц, c_0 — корневой контейнер, содержащий все базы данных, и $C = DB \cup TAB \cup \{c_0\}$ — множество сущностей-контейнеров, при этом множества контейнеров не пересекаются друг с другом и с множеством сущностей-объектов;

S — множество субъект-сессий пользователей, при этом $O \cap S = \varnothing$ и $C \cap S = \varnothing$, U — множество учетных записей и $E = O \cup C \cup S \cup U$ — множество сущностей;

$(L, \leq)$ — решетка упорядоченных уровней доступа и уровней конфиденциальности, $f_e: (O \setminus O_v) \cup C \to L$ и $f_s: U \to L$ — функции, определяющие уровни конфиденциальности и доступа соответственно;

$R_r = \{alter_r, drop_r, read_r, write_r, append_r, delete_r, execute_r, create\_routine_r, create_r, create\_user_r, create\_trigger_r, create\_view_r\}$ — множество видов прав доступа; $R_a = \{read_a, write_a, append_a\}$ — множество видов доступа и $R_f = \{write_m\}$ — множество информационных потоков;

$\mathbf{R} \subseteq \mathbf{U} \times (\mathbf{C} \cup \mathbf{O}) \times \mathbf{R_r}$, $\mathbf{A} \subseteq \mathbf{S} \times (\mathbf{O} \cup \mathbf{C}) \times \mathbf{R_a}$ и $\mathbf{F} \subseteq (\mathbf{E} \setminus \mathbf{U}) \times (\mathbf{E} \setminus \mathbf{U}) \times \mathbf{R_f}$ — множества текущих прав доступа, текущих доступов и информационных потоков соответственно.

В модель дополнительно введена функция иерархии сущностей $\mathbf{H}$: $\mathbf{C} \cup \mathbf{O_\rho} \cup \mathbf{O_t} \cup \mathbf{S} \to 2^{OUC}$, показывающая, что любая сущность является либо корневой, либо иерархически подчиненной, а также тот факт, что для любой сущности $\mathbf{e} \in \mathbf{E}$ существует единственная последовательность сущностей, начинающаяся с контейнера $\mathbf{c}_0$ и заканчивающаяся сущностью $\mathbf{H(e)}$, так что каждый ее элемент содержит предыдущий.

Сущность $\mathbf{e} \in \mathbf{E}$ будем называть иерархически подчиненной контейнеру $\mathbf{c} \in \mathbf{C}$, если $\mathbf{e} \in \mathbf{H(c)}$ или существует контейнер $\mathbf{c}_1 \in \mathbf{C}$, такой, что $\mathbf{e} \in \mathbf{H(c}_1\mathbf{)}$ и $\mathbf{c}_1 \in \mathbf{H(c)}$. Отношение иерархической подчиненности обозначается «<».

Введены следующие функции, описывающие элементы политики мандатного управления доступом типа MLS:

user: $\mathbf{S \to U}$ — функция, определяющая для каждой субъект-сессии учетную запись пользователя, от имени которой она выполняется;

owner: $\mathbf{O_\rho \cup O_t \cup O_v \to U}$ — функция, определяющая для сущностей-процедур, сущностей-триггеров и сущностей-представлений учетные записи, от имени которых они были созданы;

execute_as: $\mathbf{O_\rho \cup O_t \cup O_v \to \{as\_owner, as\_caller\}}$ — функция, задающая режим выполнения для сущностей-процедур, сущностей-триггеров и сущностей-представлений и определяющая, от имени какой учетной записи будут выполняться их последовательности правил преобразования;

triggers: $\mathbf{TAB \times \{write, append, delete\} \to O_t^*}$ — функция, определяющая последовательность триггеров, связанных с контейнером-таблицей и соответствующим методом доступа;

var: $\mathbf{S \cup O_\rho \cup O_t \to 2^{O_v}}$ — функция, задающая множество переменных, соответствующих субъект-сессии, сущности-процедуре или сущности-триггеру;

operations: $\mathbf{O_\rho \cup O_t \to O\rho^*}$ — функция, определяющая последовательность правил преобразования для сущности-процедуры или сущности-триггера;

$HLS: O \times C \to \{true, false\}$ — функция, задающая наследование уровней конфиденциальности при доступе к сущностям и определенная следующим образом: $HLS(e, c) = true$, если $e < c$ или $e = c$, определено значение $f_e(c)$ и не существует контейнера $c_1 \in C$, такого, что определено значение $f_e(c_1)$ и $e < c_1 < c$. Если $HLS(e, c) = false$, то $f_e(e) \geq f_e(c)$.

В модели предполагается, что если пользователь обладает правом доступа на контейнер, то он также обладает этим правом доступа на сущности этого контейнера. При этом считается, что если для некоторой сущности $\mathbf{e} \in \mathbf{E}$ определено значение $\mathbf{f_e(e)}$, то оно определено и для любого контейнера $\mathbf{c} \in \mathbf{C}$, такого, что $\mathbf{e < c}$.

Для описания множества прав доступа на сущности, которые могут быть переданы в рамках сессии пользователя, введено отношение $\mathbf{Grant} \subseteq \mathbf{U} \times (\mathbf{C} \times \mathbf{O}) \times \mathbf{R_r}$.

Заданы правила преобразования состояний, описывающие переход СУБД из одного состояния в другое. Существенно новыми правилами преобразования являются правила: **create_**

session, **create_view**, **access_read**, **access_append**, **create_user**, **grant_right**, **create_trigger**, **access_write**, **execute_trigger**. В таблице 1 приведены примеры задания некоторых из них.

ДП-моделью MySQL называется пара $\Sigma(\mathbf{G^*}, \mathbf{O\rho})$, если выполняются условия:

Условие 1. Состоянием модели является набор $G = (U, S, E, R, A, F, H, (f_s, f_e), user, grant, execute\_as, triggers, owner, operations, var)$.

Условие 2. Задано множество правил преобразования состояний $O\rho$.

Условие 3. Переход системы из состояния в состояние выполняется в соответствии с правилами преобразования из множества $O\rho$: G^* — множество всех состояний; $G \vdash_{o\rho} G'$ — переход системы $\Sigma(G^*, O\rho)$ из состояния G в состояние G' с использованием правила $o\rho \in O\rho$; если для системы $\Sigma(G^*, O\rho)$ определено начальное состояние, то используется обозначение $\Sigma(G^*, O\rho, G_0)$.

Таблица 1. Примеры задания правила преобразования

Правило	Состояние G	Состояние G'
$create\_session(u, s)$	$u \in U,\ s \notin S$	$S' = S \cup \{s\}$, $user'(s) = u$, $f'_s(s) = f_s(u)$
$access\_read(s, e)$	$s \in S$, $e \in DB \cup TAB \cup COL$, $\exists\ c \in C \cup O$, что $e \le c$, $(f_s(user(s)) \ge f(c)$ и $HLS(e, c) = true)$ или $(f_s(user(s)) \ge f(e)$ и $HLS(e, c) = false)$; $\nexists$ $e_1 \in C \cup O$: $f_e(e_1) < f_e(e)$, $(s, e_1, \alpha) \in A$, $\alpha \in \{write_a, append_a\}$	$S' = S \cup \{s\}$, $user'(s) = u$, $f'_s(s) = f_s(u)$

Требования политики мандатного управления доступом MLS формулируются через определение *ss-* и *-свойств состояний модели. Будем говорить, что в состоянии G системы $\Sigma(G^*, O\rho)$ доступ $(s, e, \alpha) \in A$ обладает *ss*-свойством, когда $\alpha = append_a$ или $f_s(user(s)) \ge f_e(e)$. В состоянии G системы $\Sigma(G^*, O\rho)$ доступы $(s, e_1, read_a)$, $(s, e_2, \alpha) \in A$, где $\alpha \in \{write_a, append_a\}$, обладают *-свойством, если $f_e(e_1) \le f_e(e_2)$.

В рамках построенной ДП-модели MySQL доказано следующее утверждение, аналогичное базовой теореме безопасности модели Белла — ЛаПадула [1].

Теорема. Пусть начальное состояние G_0 системы $\Sigma(G^*, O\rho, G_0)$ является безопасным в смысле модели Белла — ЛаПадулы и $A_0 = F_0 = \varnothing$, тогда система $\Sigma(G^*, O\rho, G_0)$ безопасна в смысле модели Белла — ЛаПадулы.

Основные элементы реализации механизма мандатного управления доступом

Механизм мандатного управления доступом реализован на базе MySQL версии 5.5.16 с использованием разработанной ДП-модели. Принятие решения о разрешении доступа субъект-сессии к сущности принимается мандатным механизмом после проверки соответствующих прав доступа штатным механизмом управления доступом. При этом мандатный механизм выполняет следующие проверки:

1) Проверку типа операции (чтение, запись, добавление, удаление) и соответствующего ей вида доступа, уровень доступа учетной записи пользователя и уровень конфиденциальности сущности. Данные проверки обеспечивают выполнение требований **ss**-свойства.

2) Проверку невозможности реализации информационного потока «сверху вниз» в рамках сессии пользователя. Данные проверки обеспечивают выполнение требования *-свойства.

Рассмотрим порядок функционирования мандатного механизма управления доступом в рамках СУБД MySQL. Метки безопасности, назначаемые сущностям, хранятся в служебной базе данных *mysql_mac*. Метка безопасности может принимать значение от 0 до 255. Все метки загружаются и хранятся в оперативной памяти через объекты классов *MAC_LABEL*, *MAC_DB*, *MAC_TABLE*, *MAC_COLUMN*, *MAC_EVENT*, *MAC_ROUTINE*, создаваемые на основе данных из *mysql_mac*, что позволяет сохранить уровень производительности первоначальной СУБД. Для загрузки меток безопасности учетных записей пользователей используется уже существующий класс *ACL_USER*. При этом и в том, и в другом случае для чтения меток из служебной базы данных *mysql_mac* используются собственные функции MySQL, а для назначения меток добавленная функция *mac_reload*().

Мандатное управление доступом реализуется с помощью следующих функций:

- *mac_find_type*() — определяет вид доступа по запрашиваемым субъект-сессией у дискреционного монитора безопасности правам доступа к сущности;
- *mac_ilist_label*() — определяет метки безопасности сущностей, содержащихся в дереве запроса;
- *mac_find_max_label*() — вычисляет максимальное значение среди меток безопасности сущности и содержащих ее контейнеров;
- *mac_check_access_ssp*() — обеспечивает выполнение требований *ss-* и *-свойств.

Рассмотрим порядок выполнения и механизмы реализации основных функций мандатного управления доступом:

1. После разбора SQL-запроса вызывается собственная функция дискреционного управления доступом **check_access**(), которая производит проверку прав доступа субъект-сессии к сущности, участвующей в этом запросе.

2. Вызываются функции **mac_find_max_label**(), **mac_ilist_label**() и **mac_find_type**(), определяющие необходимые метки безопасности.

3. После получения меток происходит их передача вместе с идентификатором субъект-сессии в функцию **mac_check_access_ssp**(). Первым действием после этого будет конвертация переданных меток в тип **unsigned integer** с целью их последующего корректного сравнения. В зависимости от вида запрашиваемого доступа данной функцией будут выполнены различные проверки. Сначала выполняется проверка выполнения условий **ss-**свойства. После того как все условия проверены, инициализируется переменная **current_sec_label**, находящаяся в классе **security_context** и служащая меткой сущности, к которой уже реализован доступ на запись. Далее проверяется выполнение условий *-свойства.

1. Проверка возможности реализации доступа субъект-сессии к сущностям осуществляется последовательно. Например, в запросе вида «INSERT INTO ... VALUES ((SELECT ...), ...)» сначала проверяется возможность реализации доступа вида $\mathbf{append}_a$ для оператора «INSERT», а затем доступа вида $\mathbf{read}_a$ для каждого оператора «SELECT». Таким образом, после первой проверки осуществляется запись в переменную **current_sec_label**, что означает возможность субъект-сессии осуществить запись в сущность, метка безопасности которой равна **current_sec_label**. Далее, при следующем вызове этой функции происходит проверка выполнения условий *-свойства с использованием метки безопасности сущности и значения переменной **current_sec_label**.

2. В случае предоставления субъект-сессии права на выполнение запросов от имени другой субъект-сессии первая из них будет использовать метки безопасности второй во время выполнения запросов.

Таким образом, в работе построена и реализована ДП-модель MySQL. Исследование по разработке и реализации мандатного управления доступом в СУБД MySQL в настоящее время не закончено, и данную модель целесообразно считать промежуточной. В то же время уже сейчас

удалось выявить и формально описать механизмы MySQL, приводящие к реализации запрещенных информационных потоков по памяти, а также элементы, специфичные именно для данной СУБД и не отраженные в известных ДП-моделях. Одним из дальнейших направлений работы видится добавление в модель описания механизмов реализации информационных потоков по времени.

СПИСОК ЛИТЕРАТУРЫ:

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. 2-е изд., испр. и дополн. М.: Горячая линия—Телеком, 2013. — 338 с.: ил.
2. *Смольянинов В. Ю.* Правила преобразования состояний СУБД ДП-модели // Прикладная дискретная математика. 2013. № 1. С. 50—68.
3. *Смирнов С. Н.* Безопасность систем баз данных. М.: Гелиос АРВ, 2007. — 352 с.
4. *Ткаченко Н. О.* О разработке механизма мандатного управления доступом в СУБД MySQL на основе SELinux // Прикладная дискретная математика. Приложение. 2012. Вып. 5. С. 77—79.
5. Безопасность информационных технологий. Критерии оценки безопасности информационных технологий // Руководящий документ (ГОСТ Р ИСО/МЭК 15408). М.: Гостехкомиссия России, 2002. Ч. 1—3.
6. *Bishop M.* Computer Security: Art and Science. Boston: Addison-Wesley Professional. 2002. — 1084 p.

REFERENCES:

1. *Devaynin P. N.* Modeli bezopasnosti komputernikh sistem. Upravlenie dostupom b informatsionnimi potokami/ uchebnoe posobie dlay vuzov. 2.-e izd., ispr. i dopoln. M.: Goraychay liniay — Telekom, 2013. — 338 p.: ill.
2. *Smolayninov V. Y.* Pravila preobrazovaniay sostoayniy SUBD DP-modeli // Prikladnay diskretnay matematika. 2013. № 1. P. 50—68.
3. *Smirnov S. N.* Bezopasnost sistem baz dannikh — M.: Gelios ARV, 2007. — 352 p.
4. *Tkachenko N. O.* O razrabotke mekhanizma mandatnogo upravleniay dostupom v SUBD MySQL na osnove SELinux // Prikladnay diskretnay matematika. Prilogenie, 2012. Vip. 5. P.77—79.
5. Bezopasnost informatsionnikh tekhnologiy. Kriterii otsenki bezopasnosti informatsionnikh tekhnologiy // Pukovodayshiy dokument (GOST R ISO/MEK 15408). — M.: Gostekhkomissiay Rossii, 2002. Ch. 1—3.
6. *Bishop M.* Computer Security: art and science. — ISBN 0-201-44099-7, 2002. — 1084 p.