Разработанный инструмент AspectA может применяться для отладки и трассировки скомпилированных приложений для Android во время их разработки и поддержки, при этом разработчики получают возможность легко удалить всю отладочную часть программы. Полезными примерами использования данного инструмента являются также динамический анализ вредоносных приложений и реализация дополнительных механизмов защиты в уже имеющихся приложениях.

ЛИТЕРАТУРА

1. *Filman R. E. and Friedman D. P.* Aspect-oriented programming is quantification and obliviousness [Электронный ресурс] // Technical report, RIACS, 2000. URL: `http://www.riacs.edu/research/technical_reports/TR_pdf/TR_01.12.pdf`, свободный доступ (дата обращения: 9.04.2010).
2. *Стефанцов Д. А.* Реализация политик безопасности в компьютерных системах с помощью аспектно-ориентированного программирования // Прикладная дискретная математика. 2008. № 1(1). С. 94–100.
3. *Laddad R.* AspectJ in Action: Enterprise AOP with Spring Applications, 2nd edition. Greenwich, CT, USA: Manning Publications Co., 2009. 568 p.
4. `http://eclipse.org/aspectj/` — The AspectJ Project. 2013.
5. `https://sites.google.com/a/gapp.msrg.utoronto.ca/aspectc/` — Welcome to ACC: The AspeCt-oriented C compiler. 2010.
6. `http://www.android.com/about/` — Discover Android. 2013.
7. `http://code.google.com/p/dalvik/` — Dalvik. Code and documentation from Android's VM team. 2011.
8. `https://code.google.com/p/android-apktool/` — Android-Apktool. A tool for reverse engineering Android apk files. 2013.
9. `http://asm.ow2.org/asmdex-index.html` — OW2 Consortium. ASMDEX. 2012.
10. `http://eclipse.org/` — Eclipse. The Eclipse Foundation open source community website. 2013.

УДК 004.94

РАЗРАБОТКА И РЕАЛИЗАЦИЯ МАНДАТНЫХ МЕХАНИЗМОВ УПРАВЛЕНИЯ ДОСТУПОМ В СУБД MYSQL

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

Работа посвящена разработке и реализации механизмов мандатного управления доступом в изначально дискреционной системе управления базами данных MySQL с использованием формальной модели безопасности. Рассматривается реализация мандатной политики управления доступом типа multilevel security (MLS). Предлагаемый мандатный механизм реализован в составе монитора безопасности ядра MySQL и позволяет выполнить основные требования обеспечения безопасности информационных потоков, предъявляемые к защищённым компьютерным системам. Ключевыми особенностями предлагаемого подхода являются формальное моделирование политики управления доступом на основе аппарата ДП-моделей, реализация мандатного управления доступом на уровне ядра СУБД, а также обеспечение требований безопасности информационных потоков.

Ключевые слова: *компьютерная безопасность, управление доступом, информационные потоки, формальные модели безопасности.*

В настоящее время в большинстве систем управления базами данных (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 защищённых компьютерных систем (КС). При этом используемое мандатное управление доступом, как правило, имеет следующие существенные недостатки [1 – 3]:

— отсутствует формальная модель политики управления доступом и информационными потоками;
— не учитываются особенности построения защищённых отечественных КС;
— не учитывается существенное для мандатного управления доступом требование обеспечения безопасности информационных потоков;
— мандатное управление доступом реализовано на прикладном уровне с помощью модификации или перехвата запросов SQL.

С целью исследования вопросов практической реализации мандатного управления доступом в СУБД выполнена разработка и реализация мандатного управления доступом для изначально дискреционной СУБД MySQL. Решены следующие задачи:

1) анализ изначального управления доступом;
2) разработка формальной модели политики управления доступом и теоретическое обоснование её безопасности;
3) реализация механизма мандатного управления доступом на основе построенной модели.

Анализ свойств дискреционного управления доступом СУБД MySQL проводился путём изучения документации, исследования исходного кода и выполнения компьютерных экспериментов. Особое внимание уделено идентификации информационных потоков. Известно, что анализ информационных потоков по времени в большинстве случаев выполнить сложнее, чем анализ информационных потоков по памяти [1]. Было выявлено множество способов реализации информационных потоков по времени (например, информационный поток по времени возникает при блокировании одной субъект-сессией таблицы на запись и попытке записи в неё другой субъект-сессией), но в силу технической сложности корректной реализации механизма мандатного управления доступом в данной работе рассматривается обеспечение безопасности только информационных потоков по памяти.

Установлено, что 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 запись к нескольким файлам одновременно. В СУБД MySQL субъект-сессия пользователя, как правило, не может иметь доступ на чтение и запись к двум сущностям СУБД одновременно; исключение составляют механизмы реализации запросов SQL вида

— INSERT INTO ... VALUES((SELECT...), ...);
— INSERT ... SELECT;
— UPDATE ... SET ... = (SELECT ...).

Подобные запросы могут быть использованы для реализации запрещённых информационных потоков по памяти от сущностей с высоким уровнем конфиденциальности к сущностям с низким уровнем конфиденциальности и, таким образом, нарушать требования обеспечения безопасности информационных потоков мандатной политики управления доступом.

На основе проведённого анализа управления доступом в СУБД MySQL сформулированы следующие предположения, используемые в работе:

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

Для формального моделирования и теоретического обоснования безопасности политик управления доступом и информационными потоками в СУБД MySQL строится промежуточная ДП-модель MySQL, описывающая только мандатное управление доступом типа MLS. Данная модель основана на мандатной ДП-модели [1] и СУБД ДП-модели [4]. В дальнейшем планируется существенно расширить и переработать данную модель, включив в неё элементы политик мандатного управления доступом типа TE и мандатного контроля целостности на основе модели Биба [3].

В настоящей модели используются следующие основные элементы и обозначения:

— O_p — 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, O — множество сущностей-объектов, являющееся объединением множеств всех видов, при этом множества сущностей разных видов не пересекаются;
— 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, U — множество учётных записей пользователей, при этом $S \cap O = \varnothing$ и $S \cap C = \varnothing$, $E = O \cup C \cup S \cup U$ — множество сущностей;
— $(L, \leqslant)$ — решётка 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;
— $R_r = \{alter_r, drop_r, readr, write_r, append_r, delete_r, execute_r, create\_routine_r, create_r, create\_user_r, create\_trigger_r, create\_view_r\}$ — множество прав доступа, $R_a = \{read_a, write_a, append_a\}$ — множество видов доступа, $R_f = \{write_m\}$ — множество информационных потоков;
— $R \subseteq U \times (C \cup O) \times R_r$, $A \subseteq S \times (O \cup C) \times R_a$ — множество текущих прав доступа, $A \subseteq S \times (O \cup C) \times R_a$ — множество текущих доступов, $F \subseteq (E \setminus U) \times (E \setminus U) \times R_f$ — множество текущих информационных потоков.

Функция иерархии сущностей $H : C \cup O_p \cup O_t \cup S \to 2^{O \cup C}$ отражает то, что любая сущность является либо корневой, либо иерархически подчинённой, а также то, что для любой сущности $e \in E$ существует единственная последовательность сущностей, начинающаяся с контейнера c_0 и заканчивающаяся сущностью $H(e)$ так, что каждый её элемент содержит предыдущий.

Сущность $e \in E$ называется иерархически подчинённой контейнеру $c \in C$, если $e \in H(c)$ или существует $c_2 \in C$, что $e \in H(c_2), c_2 \in H(c)$. Иерархическая подчинённость обозначается знаком $<$.

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

— $user : S \to U$ — функция, определяющая для каждого субъекта учётную запись пользователя, от имени которой она выполняется;
— $owner : O_p \cup O_t \cup O_v \to U$ — функция, определяющая для сущностей-процедур, сущностей-триггеров и сущностей-представлений учётные записи, от имени которых они созданы;
— $execute\_as : O_p \cup O_t \cup O_v \to \{as\_owner, as\_caller\}$ — функция, задающая режим выполнения для процедур, триггеров и представлений и определяющая, от имени какой учётной записи будут выполняться их последовательности правил преобразования;
— $triggers : TAB \times \{write, append, delete\} \to O_t^*$ — функция, определяющая последовательность триггеров, связанных с таблицей и соответствующим методом обработки;
— $var : S_s \cup O_p \cup O_t \to 2^{O_v}$ — функция, задающая множество переменных, соответствующих субъект-сессии, процедуре или триггеру;
— $operations : O_p \cup O_t \to OP^*$ — функция, определяющая последовательность правил преобразования для процедуры или триггера;
— $HLS : O \times C \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая наследование уровней конфиденциальности при доступе к сущностям; $HLS(e, c) = \texttt{true}$, если $e < c$ или $e = c$, определено значение $f_e(c)$ и не существует $c_1 \in C$, такого, что $e < c_1 < c$ с определённым значением $f_e(c_1)$.

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

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

Заданы правила преобразования состояний, описывающие переход СУБД из одного состояния в другое. Существенно новыми являются правила $create\_session$, $create\_view$, $access\_read$, $pass$, $access\_append$, $create\_user$, $grant\_right$, $create\_trigger$, $access\_write$, $execute\_trigger$. В таблице приведены примеры задания некоторых из них.

Примеры правил преобразования состояний

Правило	Исходное состояние	Результирующее состояние
$create\_session(u, s)$	$u \in U$, $s \notin S$	$S_s' = S_s \cup \{s\}$, $user'(s) = u$, $f_s(s)' = f_s(u)$
$create\_user(s, u, l)$	$s \in S$, $user(s) \in L_u$, $u \notin U$, $l \leqslant f_s(user(s))$, $(user(s), c_0, create\_user_r) \in R$	$U' = U \cup \{u\}$, $f_s'(u) = l$
$grant\_right(s, u, e, \alpha, grant\_option)$	$s \in S$, $u \in U$, $e \in C \cup O$, $\alpha \in R_r$, $grant\_option \in \{\texttt{true}, \texttt{false}\}$, $\exists c' \geqslant e$ $((s, c', \alpha) \in R_r)$, $\exists c \geqslant e$ $((user(s), c, \alpha) \in Grant)$	$R' = R \cup \{(u, e, \alpha)\}$, если $grant\_option = \texttt{true}$, то $Grant' = Grant \cup \{(u, e, \alpha)\}$
$access\_read(s, e)$	$s \in S, e \in DB \cup TAB \cup COL$, $\exists c \in C \cup O$ ($e < c$ или $e = c$, $f_s(user(s)) \geqslant f_e(c)$ и $HLS(e, c) = \texttt{true}$), $\nexists e_1 \in O \cup C$ ($f_e(e_1) < f_e(e)$ и $(s, e_1, \alpha) \in A$), где $\alpha \in \{write_a, append_a\}$	$A' = A \cup \{(s, e, read_a)\}$, $F' = F \cup \{(e, s, write_m)\}$

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

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

Теорема 1. Пусть начальное состояние G_0 системы $\Sigma(G^*, OP, G_0)$ является безопасным в смысле Белла — ЛаПадулы и $A_0 = F_0 = \varnothing$. Тогда система $\Sigma(G^*, OP, G_0)$ безопасна в смысле Белла — ЛаПадулы.

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

— проверку типа операции (чтение, запись, добавление, удаление) и соответствующего ей вида доступа; уровня доступа учётной записи пользователя и уровня конфиденциальности сущности, что обеспечивает выполнение требований ss-свойства ДП-модели MySQL;
— проверку невозможности реализации информационного потока «сверху вниз» в рамках сессии пользователя, что обеспечивает выполнение требования *-свойства ДП-модели MySQL.

Рассмотрим порядок функционирования мандатного механизма управления доступом, реализующего политику MLS 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, $MAC\_ROUTINE$, создаваемых на основе данных из $mysql\_mac$, что позволяет сохранить уровень производительности первоначальной СУБД. Для загрузки меток безопасности учётных 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 следующими функциями:

— $mac\_find\_type()$ — определяет вид доступа по запрашиваемым субъектом-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 её контейнеров;
— $mac\_check\_access\_ssp()$ — обеспечивает выполнение требований ss- и *- свойств.

Рассмотрим порядок выполнения и механизмы функционирования 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 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 проверяются условия *-свойства.

4) Проверка возможности реализации доступа субъект-сессии к сущностям осуществляется последовательно. Например, в запросе вида INSERT INTO ... VALUES ((SELECT ...), ...) сначала проверяется возможность реализации доступа вида $append_a$ для оператора INSERT, а затем доступа вида $read_a$ 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 $current\_sec\_label$. При следующем вызове этой функции происходит проверка условий *-свойства с использованием метки безопасности сущности и значения переменной $current\_sec\_label$.

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

Таким образом, предложена формальная модель безопасности логического управления доступом и информационными потоками (ДП-модель) для СУБД MySQL. Данная модель дополнена элементами мандатного управления доступом, обеспечивающими безопасность информационных потоков. На основе разработанной формальной модели реализован мандатный механизм управления доступом на уровне монитора безопасности ядра СУБД MySQL. Данное решение может быть использовано для построения защищённых СУБД и КС с высоким уровнем доверия к их безопасности. На основе данной работы планируется разработать формальную модель и мандатные механизмы защиты, реализующие другие мандатные политики (например, type enforcement, ABAC), учитывающие информационные потоки по памяти для всех операторов SQL, а также основные информационные потоки по времени.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
2. *Смирнов С. Н.* Безопасность систем баз данных. М.: Гелиос АРВ, 2007. 352 с.
3. *Bishop M.* Computer Security: art and science. Addison-Wesley Professional, 2002. 1084 p.
4. *Смольянинов В. Ю.* Правила преобразования состояний СУБД ДП-модели // Прикладная дискретная математика. 2013. № 1. С. 50–68.