О методах оптимизации механизмов разграничения доступа, основанных на логико-языковых средствах

О. О. Андреев

Ìîñêîâñêèé ãîñóäàðſòâåííûé óíèâåðſñèòåò èì. Ì.Â. Ëîìîíîſñîâà,
Èſſòèòóò ïðîáëåì èſôîðìàöèîííîé áåçîſàſſíîſſòè Ìîſêîâſêîãî ãîſóäàðſſòâåſſíîãî
óſſèâåðſſèſſåſſà èſſ. Ì.Â. Ëîſſîſſíîſſîſſâà, 119192, Ìîſſêâà, Ðîſſſſèÿ

Описан язык задания моделей логического разграничения доступа. Рассмотрены вопросы разработки механизмов разграничения доступа, основанных на предложенном языке, в частности способы оптимизации таких механизмов.

The article presents a language for describing logical access control policies. Problems of creating access control mechanisms based on presented language, including methods of optimizing such mechanisms are discussed.

Введение. В последние годы возрос интерес к средствам описания политик информационной безопасности информационно-вычислительных комплексов, в частности к моделям логического разграничения доступа к ресурсам этих комплексов, что обусловлено их широким использованием для решения практически значимых задач. Доступ к подобным ресурсам, к которым относятся информационные активы, средства вычислительной техники и коммуникаций, далее для краткости будем называть доступом. Средства логического разграничения доступа (далее – средства разграничения доступа), основанные на этих моделях, являются одними из главных компонент политики безопасности, включающей также средства идентификации (аутентификации) пользователей и действующих от их имени процессов, шифрования и иные компоненты [1]. Механизмы разграничения доступа могут быть встроены как в операционные системы (ОС) и системные сервисы, так и в сервисы прикладного уровня.

В программных системах, основанных на современных ОС, таких как Linux и Windows, используются механизмы разграничения доступа, основанные на таких разработанных в 70-х гг. XX в. моделях, как дискреционная [2] или многоуровневая [3]. Дискреционная модель представляет собой достаточно примитивную, хотя и простую для описания модель логического разграничения доступа, политики безопасности на основе которой получаются громоздкими в представлении, сложноверифицируемыми и неудобными для управления. Мандатная модель разграничения доступа на основе упорядоченных меток безопасности, наоборот, является слишком жесткой с точки зрения условий, которые она реализует, удобной в верификации, простой в представлении и настройке, однако пригодной для реального применения в политиках безопасности весьма узкого класса информационных систем. Реализация политик безопасности, использующих современные и более сложные модели разграничения доступа, такие как ролевая [4], сопряжена с большими трудностями внедрения дополнительных механизмов в ядра операционных систем и их использованием в составе программных комплексов.

Существенным недостатком средств описания моделей разграничения доступа в отдельных компонентах программной системы является их гетерогенность. Механизмы разграничения доступа, встроенные в различные компоненты, зачастую обладают собственным языком описания набора правил, в соответствии с которыми доступ будет разрешаться или запрещаться. Это затрудняет проверку пользователем, отвечающим за информационную безопасность системы (в дальнейшем именуемым офицером безопасности), или администратором соответствия механизмов разграничения доступа, которые реализуются в отдельных компонентах сложно организованных систем, политике безопасности системы в целом.

Указанные выше и ряд других недостатков эксплуатируемых в настоящее время систем стимулируют работы по созданию новых логико-языковых средств описания моделей разграничения доступа, таких как eXtended Access Control Markup Language (XACML) [5], Enterprise Privacy Authorisation Language [6]. Данные средства предоставляют офицеру безопасности возможность самостоятельно определить модель разграничения доступа, наиболее подходящую под нужды защищаемого информационно-вычислительного комплекса, или выбирать модель, соответствующую требованиям информационной безопасности, из моде-

лей, предлагаемых другими разработчиками. Такие логико-языковые средства обладают большим набором выразительных средств для описания условий, при которых разрешается или запрещается доступ, что является важным преимуществом этих языков перед другими способами задания моделей логического разграничения доступа. Однако следует отметить, что постоянно выполняемые при функционировании информационной системы проверки на разрешение или запрещение доступа при использовании сложных моделей логического разграничения доступа, основанных на такого рода языковых средствах, требуют существенно больших затрат вычислительных ресурсов, чем при использовании более примитивных моделей. Поскольку это приводит к замедлению работы информационной системы, одной из задач, важных для успешного внедрения и использования подобных средств в реальных информационно-вычислительных комплексах, является оптимизация механизмов разграничения доступа, основанных на выразительных логико-языковых средствах.

В настоящей работе описывается ряд подходов, позволяющих оптимизировать механизмы разграничения доступа как алгоритмического, так и программного характера. Отмечаются преимущества и недостатки представленных подходов, проводятся эксперименты по измерению производительности процедур, реализующих разграничение доступа.

Характеристика разработанного языка описания моделей разграничения доступа. В качестве основы для разработки языка описания моделей разграничения доступа будем использовать язык XACML как наиболее популярный и исследуемый в настоящее время. Этот язык, являющийся декларативным, основанным на XML языком описания моделей логического разграничения доступа, стандартизован международной организацией OASIS (Organization for the advancement of structured information standards) (http://www.oasis-open.org/). В настоящее время имеется большое количество теоретических исследований XACML, ведутся работы по практическому внедрению механизмов, основанных на этом языке, в существующие системы (http://sec.cs.kent.ac.uk/permis/) [7]. Широкое распространение языка XACML обусловлено богатством выразительных средств, позволяющих задавать широкий спектр моделей разграничения доступа, в том числе такие распространенные модели, как дискреционная и многоуровневая. Еще одним преимуществом данного языка, обеспечившим его популярность, в том числе в научных кругах, является декларативность, позволяющая упростить анализ свойств моделей разграничения доступа, заданных с помощью XACML [8, 9].

Несмотря на перечисленные преимущества, XACML является излишне усложненным для изучения в рамках исследовательских работ, в которых широта описательных возможностей, введенная в язык для облегчения интеграции с существующими системами и механизмами, является избыточной и загромождающей исследование несущественными деталями. Кроме того, с точки зрения рядового пользователя, синтаксис XACML является фактически нечитаемым.

Указанные недостатки приводят к необходимости разработки нового языка, в большой степени совместимого с XACML по семантике, однако более простого в анализе и использовании. Для обеспечения совместимости с другими языками возможна разработка автоматизированных средств преобразования моделей разграничения доступа, задаваемых на новом языке, в описание на XACML.

Как и большинство моделей, модели разграничения доступа, описываемые предлагаемым языком, базируются на трех основных понятиях: субъект, объект и доступ. Основной задачей подсистемы разграничения доступа является выдача ответов на запросы "может ли конкретный субъект получить желаемый доступ к данному объекту". Дополнительными (смежными) задачами, решение которых может поддерживаться в процессе эксплуатации системы, являются протоколирование удавшихся или неудавшихся попыток доступа и (или) предоставление данных системе обнаружения вторжений (IDS — Intrusion Detection System) и др.

Решение вопроса о разрешении или запрещении доступа подсистемой разграничения может основываться на различных данных. В случае дискреционной модели такое решение принимается на основе значений идентификаторов субъекта, объекта и доступа. В случае многоуровневой модели разграничения доступа решение является результатом анализа меток безопасности (так называемых уровней секретности в модели Белла − ЛаПадула или уровней целостности в модели Биба) субъекта и объекта, а также на типе доступа. Предлагаемый язык описывает класс моделей, являющихся, в некотором смысле, расширением принципов, положенных в основу многоуровневой модели логического разграничения доступа.

Основным понятием, на котором базируются модели разграничения доступа, задаваемые с помощью созданного автором данной работы языка, является понятие атрибута безопасности, в дальнейшем называемого просто атрибутом. Каждый объект и субъект может иметь некоторые задаваемые пользователем или системой атрибуты. Доступ разрешается или запрещается в соответствии со значением, которое выдается

заданной в конкретной модели булевозначной функции, называемой функцией доступа от атрибутов субъекта, объекта, запрашиваемого доступа и окружения. Под окружением понимается некоторый стандартный набор атрибутов, таких как "текущее время", общих для всех субъектов и объектов. Таким образом, для описания модели разграничения доступа требуется определение множества атрибутов и задание булевозначной функции от этих атрибутов, а для задания конкретной политики разграничения доступа, основанной на данной модели, необходимо задать значения атрибутов для конкретных объектов и субъектов. Примерами атрибутов служат данные о том, какую должность занимает пользователь, каков уровень секретности у запрашиваемого объекта или каково время последнего обращения пользователя к объекту. Таким образом, в качестве атрибута может записываться некоторое свойство объекта или субъекта, которое в соответствии с принятой политикой безопасности влияет на разрешение доступа. Функция доступа является переложением на формальный язык правил разграничения доступа, присутствующих в политике. Следует отметить, что для описания подобного класса моделей разграничения доступа иногда используется термин ABAC (Attribute Based Access Control), который, однако, не получил такого широкого распространения, как, например, RBAC, а данный класс не описан формально в существующих работах.

В языке, предлагаемом автором данной работы, атрибуты могут иметь следующие типы:

− булевский;

− целочисленный;

− вещественный;

− строковый;

− множество значений определенного типа (одного из вышеперечисленных или типа "множество"). В отличие от XACML, в разработанном языке атрибуты по умолчанию не могут иметь несколько значений одновременно, для этого они должны иметь тип "множество".

Традиционные модели разграничения доступа являются в основном "статическими", т. е. решение о предоставлении субъекту определенного доступа к объекту постоянно или зависит от времени, в которое доступ совершается, и меняется лишь при смене политики разграничения доступа. Такое ограничение, как правило, не позволяет реализовывать востребованные в современных информационных системах политики разграничения доступа, в которых при разрешении или запрещении доступа должна учитываться предшествующая история обращений к объекту. Примером такой политики может служить запрет на доступ к какому-либо ресурсу более трех раз в течение дня.

Разработанный язык включает средства задания политик разграничения доступа, которые могут иметь так называемые пост-действия [10] – последовательность действий, изменяющих атрибуты участвующих в попытке доступа субъекта и объекта. Примером пост-действия является увеличение атрибута "количество доступов" на единицу при каждом доступе. Такой атрибут может использоваться для запрещения доступа при превышении некоторого порогового значения.

В рассматриваемом языке представление модели разграничения доступа структурно подразделяется на модели и правила. Правило представляет собой базовую единицу модели, состоящую из области применения, условия и результата.

Область применения правила состоит из четырех булевых предикатов, описывающих допустимые субъекты, объекты, типы доступа и окружение, к которым это правило можно применить. Предикаты зависят от атрибутов субъекта, объекта, от типа доступа или окружения соответственно. В случае если предикат отсутствует, предполагается, что правило может быть применено к любому субъекту, объекту, типу доступа и окружению, в зависимости от того, какой из предикатов отсутствует.

Условие является булевым предикатом, который определяет результат применения правила к запрашиваемому доступу. Условие зависит от атрибутов субъекта, объекта, типа доступа и окружения. В случае отсутствия такого условия считается, что оно всегда верно.

Область применения и условие могут быть построены из логических связок, базовых предикатов, атрибутов субъектов, объектов, типа доступа и окружения и константных значений. В случае если используемый атрибут отсутствует, его значением считается специально выделенное значение nil. В случае несовпадения типов в базовых предикатах (например, сравнение атрибута, имеющего целочисленное значение и константы строкового типа) считается, что правило неприменимо к запросу. Результатом применения правила может быть либо "разрешено", либо "запрещено".

Алгоритм вычисления применения правила к запросу на доступ имеет следующий вид:

1) проверяется область применения правила; в случае если субъект, объект, тип доступа или окружение не подходят под область применения, правило считается неприменимым;

2) в случае если правило применимо, проверяется условие; если условие истинно, то правило возвращает свой результат, в противном случае правило возвращает отрицание результата (т. е. "запрещено" для "разрешено", и наоборот).

Модель состоит из области применения, набора правил и других моделей, алгоритма объединения правил и пост-действий.

Алгоритм комбинирования задает способ, которым будут объединяться результаты применения правил к вопросу о разрешении доступа. Алгоритм может быть "приоритет разрешения" или "приоритет запрещения".

Пост-действия являются последовательностями присваиваний атрибутам функций других атрибутов и константных значений. Для каждой модели может быть определено два пост-действия: для случая, если модель разрешает доступ, и для случая, если модель не разрешает доступ.

Результат применения модели к запросу на доступ вычисляется следующим образом:

1) проверяется область применения модели; в случае если субъект, объект, тип доступа или окружение не подходят под область применения, модель считается неприменимой;

2) к запросу последовательно применяются все правила и модели;

3) если ни одно из правил или моделей неприменимо, то модель также считается неприменимой;

4) если алгоритм комбинирования модели – "приоритет разрешения", то в случае существования хотя бы одного правила или модели, разрешающих доступ, модель разрешает доступ, в противном случае доступ запрещается;

5) если алгоритм комбинирования модели – "приоритет запрещения", то в случае существования хотя бы одного правила или модели, запрещающих доступ, модель запрещает доступ, в противном случае доступ разрешается.

Если модель применима к запросу на доступ, в зависимости от результата выполняется соответствующее пост-действие, если оно было определено.

Область применения соответствует понятию Target языка XACML, условие — понятию Condition, результат — понятию Effect, пост-действие — понятию Obligation, модель — понятию Policy.

Ниже приводится пример модели разграничения доступа, описанный на рассматриваемом языке.

```
model UniversityAccess: {
    model StudentAccess: {
          description: 'Эта модель регламентирует доступ
                студентов к информационным ресурсам',
          rule: {
                description: 'Это правило разрешает студентам
                      читать учебные материалы в учебное время'
                target: {
                      subject: status == 'student',
                      object: type == 'textbook',
                      access: type == 'read',
                      environment: timeofday > 9h00m
                            and timeofday < 18h00m
                },
                result: grant
          }
    }
    model ProfessorAccess: {
          description: 'Эта модель регламентирует доступ
                профессоров к информационным ресурсам',
          rule: {
                description: 'Это правило разрешает профессорам
                      доступ к любым ресурсам'
                target: {
                      subject: status == 'professor'
                },
```

```
            result: grant
        }

}
```

Приведенный пример содержит описание простой модели, которая запрещает студентам доступ к чему-либо, кроме учебных материалов, в определенное время, разрешает профессорам любой доступ к любым информационным ресурсам и запрещает любой другой доступ.

Методы оптимизации механизмов разграничения доступа на основе разработанного языка. Широкая выразительность способа описания моделей логического разграничения доступа тесно связана с вопросами производительности механизмов безопасности, основанных на таком способе. Использование сложных моделей, состоящих из десятков и даже сотен правил, может существенно замедлить работу информационного ресурса, особенно если проверки на разрешение доступа должны проводиться часто. Ниже описываются основные методы оптимизации вычисления, разрешен или не разрешен доступ, которые могут использоваться в основанных на разработанном языке механизмах как алгоритмического, так и программного рода. Приведенные оптимизации не зависят друг от друга и в зависимости от профиля использования конкретного механизма разграничения доступа могут применяться как совместно, так и по отдельности.

Эквивалентные преобразования моделей разграничения доступа. Одним из практически значимых классов оптимизации механизмов разграничения доступа является эквивалентное преобразование моделей, т. е. преобразование одной модели разграничения доступа в другую – такую, что все доступы, разрешаемые (запрещаемые) первой моделью, разрешаются (запрещаются) и второй моделью, при этом выполняются необходимые пост-действия. При использовании подобного класса оптимизаций механизм разграничения доступа производит необходимые преобразования модели разграничения доступа на этапе инициализации (загрузки) и в дальнейшем использует преобразованную модель для решения вопроса о предоставлении доступа.

Специфика механизмов логического разграничения доступа заключается в том, что типичная сложная модель разграничения доступа, описанная с помощью языка, состоит из тысяч или даже десятков тысяч правил, каждое из которых может быть применимо к запросу на предоставление доступа. В случае стандартной, "наивной" реализации механизмов безопасности большая часть времени для обработки запроса при использовании такой сложной модели тратится на поиск правил, которые могут быть применены к этому запросу. С учетом сказанного выше в целях оптимизации представляется целесообразным рассмотрение таких способов преобразования моделей разграничения доступа, которые помогут максимально сократить затраты на поиск правил, подходящих под запрос. Одним из классов эквивалентных преобразований, которые можно использовать для такой оптимизации, является вынесение общей области применимости. Это преобразование состоит в добавлении к области определения модели некоторых ограничений, которые выводятся из ограничений, содержащихся в данной модели правил и других моделей. Корректность такого преобразования основывается на следующем утверждении: при добавлении в область применимости модели объединения условий всех правил и моделей, содержащихся в ней, или более слабого условия, получаемая модель будет эквивалентна исходной. Справедливость данного утверждения следует из алгоритма применения модели к запросу. Действительно, любой запрос, к которому может быть применима модель, должен входить в область применимости одного из правил или моделей, содержащейся в ней. Таким образом, к запросу применима и преобразованная модель. Если же запрос был неприменим, то он останется неприменимым, так как области применимости правил не изменились. Следовательно, указанный класс преобразований действительно является классом эквивалентных преобразований.

Вторым классом эквивалентных преобразований является разделение моделей таким образом, что одна модель разграничения доступа делится на две модели, области применимости которых при объединении дают область применимости изначальной модели или более широкую. После этого из получившихся моделей устраняются содержащиеся в них правила и модели, область применимости которых не пересекается с областью применимости полученных моделей. Две модели, полученные в результате таких операций, "в сумме" эквивалентны исходной модели. Доказательство эквивалентности данных преобразований аналогично доказательству для предыдущего случая.

Указанные эквивалентные преобразования могут применяться в задачах оптимизации вычисления, разрешенных следующим образом. Специально подобранное вынесение области применимости из отдельных правил в модели и разбиение на подмодели позволит сократить число проверок. Вместо последовательности проверок на применимость каждого отдельного правила будет происходить одна проверка на применимость

модели, и в случае ее неудачи все остальные проверки будут игнорироваться. В некотором смысле при построении таких эквивалентных моделей происходит преобразование линейного списка правил в дерево, поиск по которому более быстрый, чем поиск по списку.

Приведем достаточно простой алгоритм оптимизации моделей разграничения доступа, основанный на двух приведенных выше классах преобразований и состоящий из двух основных шагов:

1) получение области допустимых значений для каждого атрибута субъекта, объекта или доступа;

2) разделение на подмодели по принципу принадлежности атрибута диапазону значений.

Опишем более подробно первый шаг. Для области применимости каждого правила в заданной модели проводится определение возможных значений каждого атрибута субъекта, объекта и окружения. Данная операция может быть описана следующим рекурсивным алгоритмом, начинающимся с применения к каждому из трех булевых предикатов, описывающих области применимости к субъектам, объектам и окружению:

1) в случае если предикат представляет собой логическую связку "или" двух предикатов, в каждом из них определяется диапазон значений атрибутов и эти диапазоны объединяются для каждого атрибута по отдельности;

2) в случае если предикат представляет собой логическую связку "и" двух предикатов, в каждом из них определяется диапазон значений атрибутов и эти диапазоны пересекаются для каждого атрибута по отдельности;

3) в случае если предикат представляет собой сравнение атрибута с константой или операции над множествами, такие как проверка на принадлежность множеству, включение в множество и подобные, диапазон значений для данного атрибута считается в соответствии с предикатом;

4) в иных случаях диапазон атрибутов считается любым.

В качестве примера рассмотрим правило со следующей областью применимости:

```
target: {
          subject: (time < 18h00m and role == 'student')
                or role == 'professor'
}
```

Для данного правила область возможных значений атрибутов субъекта, полученных с помощью приведенного алгоритма, имеет следующий вид: role in ('student', 'professor'). Областью возможных значений объекта являются все значения.

Если взять декартово произведение областей возможных значений для каждого из атрибутов субъекта, объекта и окружения, т. е. объединить все полученные условия на отдельные атрибуты логической связкой "и", то получится область применимости, более широкая или совпадающая с объединением областей применимости всех правил. Таким образом, можно применить преобразование первого класса — вынос полученного условия на область применимости в область применимости модели. Данная операция производится рекурсивно, начиная с моделей нижнего уровня вложенности.

После завершения первого шага основного алгоритма структура модели разграничения доступа сохраняется, однако к каждой модели добавляется дополнительное условие на область ее применимости.

Второй шаг заключается в разделении областей возможных значений атрибутов на диапазоны и соответствующем разделении преобразованных на первом шаге моделей на отдельные подмодели. Для выполнения такого преобразования область возможных значений каждого атрибута субъекта, объекта или окружения разбивается на связные подобласти, т. е. на непересекающиеся области значений, которые отделены друг от друга. Например, область значений ('professor', 'student') разбивается на ('professor') и ('student'), а $(0,\ 1)\cup(2,\ 3)$ разбивается на (0, 1) и (2, 3). Такое правило разбиения является чисто эмпирическим и следует из анализа существующих моделей логического разграничения доступа. После разбиения областей значений каждого из атрибутов на подобласти рассматриваются все возможные сочетания подобластей по всем атрибутам (в случае если этих подобластей было n_1 для первого атрибута, n_2 для второго, ..., n_k для последнего атрибута, число таких сочетаний равно $n_1 \times n_2 \times \ldots \times n_k$). При этом модель разграничения доступа разбивается на соответствующее количество подмоделей, для каждой из которых имеется сочетание подобластей значений атрибутов. Заметим, что такое разбиение на максимально возможное число моделей не приводит к существенному увеличению объема памяти, занимаемого описанием модели. Это обусловлено тем, что все преобразованные модели используют один и тот же набор правил, содержащийся в исходной модели.

Следует отметить возможность оптимизации программного характера, связанной с тем, как порождается модель при применении указанного выше способа оптимизации. Чтобы понять ее суть, рассмотрим частный случай двух атрибутов. Пусть имеются два атрибута субъекта a_1 и a_2 с областями допустимых значений $T_1 \cup T_2$ и $T_3 \cup T_4$ соответственно, где T_1,..., T_4 – подобласти значений. При этом $T_1 < T_2$ и $T_3 < T_4$, т. е. все значения из T_1 меньше, чем значения T_1, и аналогичное верно для двух других подобластей. В этом случае из изначальной модели M получается четыре модели M_{13}, M_{14}, M_{23}, M_{24} для четырех пар подобластей. Эти модели можно представить в виде дерева поиска, в котором на первом шаге при получении запроса на доступ проверяется принадлежность значения атрибута a_1 одной из двух областей значений, на втором шаге – принадлежность значения атрибута a_2 одной из двух других областей. Такая программная оптимизация позволяет дополнительно ускорить процесс поиска правил, удовлетворяющих запросу на доступ.

Кеширование разрешенных доступов. Одним из способов оптимизации, наиболее часто применяемых в современных информационных системах, является кеширование результатов вычислений. Кеш – это ассоциативный массив, который ставит в соответствие запросу на вычисление результат, соответствующий ему. Подобная оптимизация в механизмах разграничения доступа наиболее эффективна в случаях, когда доступ субъекта к объекту происходит существенно более часто, чем изменение политики разграничения доступа. Под изменением политики понимается изменение модели разграничения доступа либо атрибутов субъекта или объекта. При таких условиях кеширование позволяет заменить повторные проверки сложных условий при запросах на доступ простым поиском по кешу, который выполняется очень быстро. Примером ситуации, когда кеширование дает значительные преимущества, может быть последовательное чтение пользователем данных из файла в ОС, в которой проверка на разрешение доступа проводится при каждой файловой операции. В этом случае запросы на доступ дублируют друг друга при каждой операции чтения.

Применение кеша требует определения политики кеширования. Иными словами, необходимо определить, какие именно решения о доступе будут кешироваться, что будет использоваться в качестве ключа кеша, как определять его размер и обновлять в случае изменения политики разграничения доступа.

При использовании кеширования для оптимизации механизмов разграничения доступа на основе разработанного языка существует несколько вариантов выбора ключей кеша. Одним из вариантов является использование в качестве ключа кортежа из всех значений атрибутов, которые запрашиваются в модели разграничения доступа, включая атрибуты субъекта, атрибуты объекта, тип доступа и окружение. Выбор такого ключа позволяет кешировать все запросы на доступ, так как решение о доступе зависит исключительно от значений элементов кортежа. Вместе с тем следует отметить, что его применение требует большого объема памяти для хранения ключей и замедляет поиск по кешу.

Более целесообразным является использование в качестве ключа тройки (идентификатор субъекта, идентификатор объекта, тип доступа), которое налагает некоторые ограничения на механизм кеширования. Во-первых, подсистема обеспечения безопасности должна иметь уникальные идентификаторы субъектов и объектов, при этом оптимальным является вариант, когда эти идентификаторы имеют целочисленные значения (примерами таких идентификаторов могут служить идентификатор пользователя `uid`, номер файла `inode` на файловой системе в Unix). Во-вторых, в случае использования такого ключа не могут быть кешированы результаты применения правил или подмоделей, которые зависят от окружения. Независимо от выбора ключа в кеш должен помещаться не только результат (разрешение или запрещение доступа), но и ссылка на список выполненных пост-действий. При повторном запросе на доступ, имеющемся в кеше, эти пост-действия должны быть выполнены повторно.

Вопрос определения размера кеша в большинстве случаев может быть решен лишь эмпирическим путем. Использование кеша малого размера позволяет сохранять информацию только о нескольких последних доступах, однако это ускоряет поиск решения о доступе по кешу и уменьшает объем используемой механизмом разграничения доступа памяти. Использование кеша большого размера позволяет сократить число "полных" проверок на доступ, однако требует большего объема памяти и увеличивает среднее время доступа к уже закешированным элементам. Настройка размера кеша должна производиться отдельно для каждого случая, однако, как правило, целесообразно использование кеша небольшого размера.

Процесс обновления политики разграничения доступа является достаточно простым:

1) при каждом изменении политики разграничения доступа весь кеш должен быть очищен;

2) при изменении атрибутов субъекта или объекта из кеша необходимо удалить все записи, ключ которых содержит измененный субъект или объект.

Одним из важных этапов реализации кеширования является выбор структуры данных. Существует несколько вариантов структур включая следующие:

– дерево поиска;

– линейный массив, в том числе упорядоченный;

– хеш-таблица.

Использование дерева поиска в кеше механизма разграничения доступа не очень целесообразно, так как достаточно велики расходы на построение и поддержание дерева. В худшем случае, если последовательность доступов не повторяется, добавление новых значений и удаление старых из кэша, основанного на дереве поиска, требуют существенных и при этом бесполезных затрат вычислительных ресурсов. Это относится и к использованию упорядоченного линейного массива. Использование простого линейного массива позволяет сократить расходы на добавление (обновление) элементов в кеше, однако замедляет поиск по кешу. Наиболее целесообразным представляется использование в механизмах безопасности кеша на основе хеш-таблицы с фиксированным размером списков для хеш-коллизий. Такой кеш позволяет сократить вычислительные затраты на поиск значений и добавление (обновление) элементов и является оптимальным как для случая большого числа идентичных запросов на доступ, так и для случая, когда запросы на доступ не повторяются.

Преобразование в исполняемый код. Генерация исполняемого кода является наиболее "экстремальным" способом оптимизации, который следует использовать только в тех случаях, когда необходима максимальная производительность и другие способы не дают искомого результата, например при внедрении механизмов разграничения доступа в ядро ОС на основе языка его описания.

При использовании генерации исполняемого кода модель разграничения доступа при ее установке преобразуется в исполняемый код под архитектуру процессора системы. После этого во время работы системы, предоставляющей доступ, вызывается сгенерированный код. Для генерации исполняемого кода могут быть применены различные способы, в частности генерация исходного кода на одном из компилируемых языков, например на языке C, с последующей компиляцией полученного кода в машинный.

В настоящей работе рассмотрен способ генерации исполняемого кода с помощью виртуальной машины Low Level Virtual Machine (LLVM) (http://llvm.org/), которая является инфраструктурой для разработки компиляторов, содержащей широкие возможности для оптимизации получаемого кода. Эта инфраструктура основана на специализированном низкоуровневом представлении Internal Representation (IR), не "привязанном" ни к высокоуровневым языкам программирования, ни к ассемблерам для конкретной архитектуры процессора. При работе с LLVM верхним уровнем являются различные компиляторы, генерирующие код в форме IR. После генерации IR-кода LLVM производит оптимизирующие преобразования и генерирует машинный код. Это позволяет не писать генератор ассемблера для каждой целевой архитектуры процессора, а также использовать широкие возможности по оптимизации, предоставляемые LLVM. Следует отметить, что в отличие от таких распространенных виртуальных машин, как JVM или CLI, LLVM является достаточно низкоуровневой виртуальной машиной, которая не поддерживает какую-либо объектную семантику, не имеет встроенных механизмов сборки мусора и не навязывает среду исполнения. Последнее обстоятельство особенно важно при применении LLVM в механизмах безопасности.

Для дальнейшего изложения рассмотрим более подробно особенности промежуточного представления. Система команд промежуточного представления IR состоит из небольшого числа команд, оперирующих с базовыми типами, включающими целочисленные, вещественные, структурные типы и массивы. К числу операций относятся арифметические операции, операции по передаче управления, включающие вызов функций, операции выделения памяти.

В LLVM отсутствуют встроенные строковые типы и более сложные типы данных, такие как списки или деревья. Операции с такими типами необходимо реализовывать с помощью базовых типов. Система типов разработанного языка описания моделей разграничения доступа является более сложной, чем у LLVM, поэтому для использования данной инфраструктуры в механизмах разграничения доступа, основанных на языке, необходимо реализовать все базовые типы языка в форме IR. Для представления строковых типов выбрана структура данных, содержащая 32-битный хеш строки и саму строку с завершающим нулем. Такая структура данных позволяет быстро сравнивать строки на неравенство. Для реализации множества выбран линейно упорядоченный массив, в начале которого записаны тип элементов и их количество. Для каждой политики к сгенерированному для нее коду в IR-представлении добавляется набор описанных в языке стандартных функций типа "проверить элемент на принадлежность множеству" или "сравнить строки". Во время шага оптимизаций LLVM встраивает (inline) такие функции в сгенерированный код.

Для получения атрибутов субъекта, объекта и окружения используются функции `subject`, `object` и `environment`, а для установки значений атрибутов в пост-действиях − функции `setsubject` и `setobject`, указатели на которые передаются механизмом безопасности сгенерированному машинному коду. Реализация этих функций зависит от конкретного способа хранения атрибутов в подсистеме безопасности. Первые две функции должны принимать на вход идентификатор субъекта или объекта соответственно, а также идентификатор атрибута и возвращать значение этого атрибута. Последние две функции должны принимать на вход идентификатор субъекта или объекта соответственно, идентификатор атрибута и значение. Соответствие имени атрибута, которое приведено в текстовом описании модели разграничения доступа, и его идентификатора формируется на этапе генерации исполняемого кода.

В результате генерации исполняемого кода получается функция, которая, в зависимости от того, разрешен или запрещен доступ, принимает на вход идентификаторы субъекта и объекта и тип действия и возвращает булевское значение. При использовании кеширования для оптимизации необходимо в код на IR дополнительно вставить операции по созданию, поддержке и поиску по кешу.

Тестирование разработанных методик оптимизации. При разработке методик оптимизации требуется тщательная проверка их целесообразности. Поскольку действительно большие модели разграничения доступа, используемые в информационных системах, в открытом режиме отсутствуют, для тестирования представленных способов оптимизации создано средство для автоматизированной генерации моделей разграничения доступа на разработанном языке, каждая из которых содержала от 100 до 10 000 правил, зависящих от 50 различных атрибутов. После генерации моделей разграничения доступа генерировался поток запросов на доступ, которые подавались на вход механизмов разграничения доступа. При этом проверялась корректность оптимизации, а именно одинаковые решения о доступе в случае базовой реализации и оптимизированной реализации, и сравнивалось время, затраченное на выдачу ответа. Базовая реализация была разработана на языке программирования C, реализация, использующая оптимизацию типа "эквивалентное преобразование модели", – на языках программирования C и Python (код, проверяющий доступ, написан на C, а код, выполняющий преобразования, – на Python). Реализация, использующая оптимизацию "кеширование запросов", также разработана на языке C, программа с применением оптимизации "преобразование в исполняемый код" – на языке программирования Python.

В качестве платформы для теста использовался компьютер с процессором "Intel Core Duo 2" и объемом оперативной памяти, равным 2 Гб. На компьютере была установлена ОС Ubuntu 8.10, версия компилятора GCC 4.3.2, версия интерпретатора Python 2.5.2.

Результаты тестирования метода оптимизации "эквивалентное преобразование модели". Для тестирования программной реализации метода оптимизации "эквивалентное преобразование моделей" генерировался поток из 10 000 случайных запросов на доступ. В табл. 1 представлены результаты сравнения производительностей базовой и оптимизированной версий механизма разграничения доступа. Видно, что производительность оптимизированной реализации на несколько порядков превышает производительность базовой реализации, причем при увеличении размера модели разграничения доступа разрыв увеличивается.

Результаты тестирования метода оптимизации "кеширование запросов". Для тестирования данной оптимизации генерировался поток из 10 000 запросов на доступ, при этом сгенерированные запросы поступали сериями по 30 идентичных запросов. В табл. 2 представлены результаты сравнения производительности базовой и оптимизированной версий механизма разграничения доступа.

Нетрудно заметить, что производительность оптимизированной реализации существенно выше производительности базовой реализации. При этом отношение производительностей близко к значению теоретически ожидаемого коэффициента, равному 30.

Таблица 1

Размер модели разграничения доступа	Время на разрешение запроса на доступ, мс	
	Базовая реализация	Оптимизированная реализация
100	534	21
1000	5249	54
10 000	52 983	90

Таблица 2

Размер модели разграничения доступа	Время на разрешение запроса на доступ, мс	
	Базовая реализация	Оптимизированная реализация
100	520	31
1000	5113	240
10 000	55 093	2100

Таблица 3

Размер модели разграничения доступа	Время на разрешение запроса на доступ, мс	
	Базовая реализация	Оптимизированная реализация
100	550	120
1000	5291	1275
10000	51004	18 320

Результаты тестирования метода оптимизации "преобразование в исполняемый код". Для тестирования реализации метода оптимизации "преобразование в исполняемый код" генерировался поток из 10 000 случайных запросов на доступ. В табл. 3 представлены результаты сравнения производительностей базовой и оптимизированной версий механизма разграничения доступа. Видно, что производительность оптимизированной реализации в несколько раз превышает производительность базовой реализации, однако ниже производительностей оптимизаций "преобразование моделей" и "кэширование запросов".

На основе представленных экспериментальных данных можно сделать вывод, что предложенные методы оптимизации механизмов разграничения доступа, основанных на сложных языковых моделях описания, позволяют существенно сократить время ответа на запрос на предоставление доступа. Все три метода оптимизации независимы друг от друга, и для достижения максимального ускорения возможно их одновременное использование.

Заключение. Рассмотрен язык описания моделей разграничения доступа и представлены основные методы оптимизации механизмов, основанных на этом языке. Путем тестирования показана возможность существенной оптимизации таких механизмов по сравнению с базовой реализацией для их использования в практически значимых информационных системах включая высокопроизводительные.

Список литературы

1. Васенин В. А. Проблемы математического, алгоритмического и программного обеспечения компьютерной безопасности в Интернет // Материалы конф. МаБИТ-03. М.: Б. и., 2004. С. 111-143.
2. A guide to understanding discretionary access control in trusted systems, NCSC-TG-003. National Computer Security Center, 1987.
3. Bell D., LaPadula L. Secure computer systems: unified exposition and multics interpretation: Tech. Report MTR-2997. S. l.: Mitre Corp., 1976.
4. Ferrariolo D., Kuhn R. Role-based access controls // Proc. of the 15th National computer security conf. Baltimor (USA), 13-16 Okt. 1992. P. 554-563.
5. eXtensible access control markup language (XACML) commitee specification. S. l.: OASIS Open, 2003.
6. Enterprise privacy authorization language: IBM Research Report. S. l.: IBM, 2003.
7. Lorch M., Proctor S., Lepro R., et al. / First experiences using XACML for access control in distributed systems // Proc. of the ACM workshop on XML security, 31 Oct., 2003.
8. Fisler K., Krishnamurthi S., Meyerovich L., Carl M. Policy verification and change impact analysis // Proc. of the workshop Ottawa "New challenges for access control", Ottawa (Canada), 27 Apr. 2005.
9. Martin E., Xie T. Automated test generation for access control policies via change-impact analysis // Proc. of the 3rd Intern. workshop on software engineering for secure systems (SESS 2007), May 2007.
10. Kudo M., Satoshi H. XML document security based on provisional authorization, 2000.

Олег Олегович Андреев − науч. сотр. Ин-та проблем информационной безопасности Московского гос. ун-та им. М.В. Ломоносова;
e-mail: olegoandreev@yandex ru