Метамодель ролевого разграничения доступа

Н.Ф. Богаченко
nfbogachenko@mail.ru

Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

Аннотация

В работе представлена объектно-ориентированная метамодель политики ролевого разграничения доступа. Основные множества и отображения описаны в терминах классов и отношений наследования и агрегации. Управление доступом (функционирование) рассмотрено с позиций метапрограммирования.

Введение

Для обеспечения функционирования крупномасштабных объектов информатизации все более востребованными становятся большие информационные системы (БИС). Потребности бизнеса и государства в интеграции разрозненных информационных систем, обслуживающих эти структуры, определяют активное развитие и расширение сферы внедрения БИС. По-прежнему главной из проблем остается сложность создания и сопровождения БИС. В области программирования общепризнанным способом борьбы со сложностью системы является объектно-ориентированный подход. Так как для подсистемы безопасности БИС упомянутая проблема сложности также актуальна, объектно-ориентированные принципы постепенно распространяются и на сферу информационной безопасности [1, 2]. В настоящее время появился термин «объектно-ориентированная безопасность», понимаемый, в частности, как практика использования общих объектно-ориентированных шаблонов проектирования в качестве механизма для защиты информации [3].

Центральной задачей системы защиты информации является обеспечение конфиденциальности данных. Наиболее действенным методом решения данной проблемы считается управление доступом к информационным ресурсам. Набор правил разграничения доступа в информационной системе, который принято называть политикой управления доступом, определяет основные принципы регулирования использования всех ресурсов системы.

В рамках подходов к реализации политики управления доступом ролевая модель рассматривается как объектно-ориентированное решение, способное понизить сложность администрирования системы с большим количеством пользователей и ресурсов. Тем не менее, не смотря на то, что идея объектно-ориентированной природы ролевой политики управления доступом не нова и заложена в эту модель изначально [4], отсутствует строгая объектно-ориентированная интерпретация ролевой политики (например в терминах объектной модели какого-либо языка программирования). Подобные идеи обсуждались в ряде работ. Так в статье [5] впервые была дана интерпретация полномочий как набора базовых классов, а ролей – как множества классов-наследников. В работе [6] авторы описывают метамодель ролевой политики управления доступом в виде иерархии классов на языке UML.

Основная направленность работ по объектной интерпретации ролевой модели заключается в программной реализации ролевого принципа управления доступом в рамках той или иной информационной системы. Вместе с тем переход на объектно-ориентированную терминологию представляет собой удобный

In: Sergey V. Belim, Nadezda F. Bogachenko (eds.): Proceedings of the Workshop on Data, Modeling and Security 2017 (DMS-2017), Omsk, Russia, October 2017, published at http://ceur-ws.org

инструмент для дальнейшей формализации ролевого подхода и получения теоретических оценок безопасности. Например, объектная модификация классических дискреционных моделей управления доступом позволила расширить класс систем, для которых доказана алгоритмическая безопасность [7], и установить взаимосвязь с мандатными моделями [8].

В данной статье для объектно-ориентированного представления ролевой политики управления доступом использованы объектная модель, допускающая множественное наследование, и механизмы метапрограммирования[1]. Использована терминология языка программирования C++. Выбор языка обусловлен, в частности, наличием в C++ возможности создания класса на основе нескольких базовых и наиболее распространенным синтаксисом.

1 Спецификация основных множеств и отображений

Базовая модель ролевой политики управления доступом предполагает задание следующих основных элементов информационной системы [10]:

- множество ролей $R = \{r_1, \ldots, r_n\}$;
- множество полномочий $P = \{p_1, \ldots, p_m\}$;
- множество пользователей $U = \{u_1, \ldots, u_s\}$;
- отображение $RP : R \to 2^P$, которое каждой роли r_i сопоставляет набор полномочий $RP(r_i) \subseteq P$;
- отображение $UR : U \to 2^R$, которое каждому пользователю u_j сопоставляет подмножество разрешенных для авторизации ролей $UR(u_j) \subseteq R$.

Управление доступом заключается в авторизации пользователя в каждом сеансе работы в системе, то есть в определении множества ролей, по которым пользователь осуществляет доступ к ресурсам в данном сеансе. Формально процесс авторизации можно описать отображением $CU : C \times U \to 2^R$, где C – множество сеансов работы в системе. При этом должно выполняться условие: $CU(c, u_j) \subseteq UR(u_j)$.

Часто, при попытке объектной интерпретации ролевой модели, роли сопоставляются с классами, полномочия – с методами классов, а пользователи – с объектами (или экземплярами) классов [2]. В данной работе предлагается несколько иная спецификация ролей, полномочий и пользователей, позволяющая в большей мере отразить особенности этих элементов.

Полномочия в общем случае объединяют в себе и операции доступа, и объекты доступа (например, запросы на обработку данных в СУБД). Поэтому целесообразно интерпретировать каждое полномочие не как метод, а как отдельный класс, инкапсулирующий поля-данные и методы их обработки. Таким образом, каждому полномочию p_k $(k = 1, \ldots, m)$ сопоставляется класс Pk.

Роль, как активная сущность, представляет собой типовой набор полномочий. Следовательно, каждой роли r_i $(i = 1, \ldots, n)$ в объектной модели следует сопоставить класс Ri.

Множество пользователей – это также набор классов: для каждого пользователя u_j $(j = 1, \ldots, s)$ в объектной модели создается класс Uj.

Для спецификации отображения RP возможны два подхода. Первый – классы-роли являются наследниками базовых классов-полномочий. Но при этом и полномочия, и роли становятся сущностями одной природы, связанными отношением наследования, которое определяется отображением RP. Второй подход позволяет учесть различия в понятиях «полномочие» и «роль». Предлагается между классами-ролями и классами-полномочиями установить отношение агрегации. Класс-контейнер – это класс-роль Ri. Полями класса Ri являются ссылки на объекты классов-полномочий, сопоставленных полномочиям из множества $RP(r_i)$. В данном случае агрегация предпочтительнее композиции, так как при композиции полномочие может быть частью одной и только одной роли, тогда как при агрегации полномочие может быть частью нескольких ролей.

Спецификация отображения UR аналогична подходу, примененному к отображению RP. Между классами-пользователями и классами-ролями вновь устанавливается отношение агрегации. На этот раз классом-контейнером является класс-пользователь Uj, а поля этого класса определяются ролями из множества $UR(u_j)$ и представляют собой ссылки на объекты классов-ролей.

[1] Метапрограммирование – парадигма программирования, построенная на программном изменении структуры и поведения программ [9].

Следующим важным моментом в построении ролевой модели является вопрос о возможном делегировании полномочий между ролями. Наиболее распространены ролевые модели с иерархической организацией системы ролей. В этом случае на множестве ролей R задается отношение частичного порядка «$\geq$», называемое отношением доминирования/подчинения. При этом доминирование одной роли над другой подразумевает полное наследование ее полномочий. Кроме того, при авторизации пользователя на какую-либо роль r_i, он получает полномочия и всех ролей, подчиненных r_i. В объектной модели эти принципы в полной мере отражены в отношении наследования между классами. Диаграмма Хассе (транзитивное замыкание диаграммы отношения порядка), порожденная отношением доминирования/подчинения на множестве ролей, задает диаграмму иерархии классов R1, ..., Rn. Листовым вершинам (вершинам без исходящих дуг) диаграммы Хассе соответствуют базовые классы-роли. Далее, используя механизм множественного наследования, порождаются производные классы-роли. Одной из проблем множественного наследования является дублирование полей общего предка. Чтобы любой класс-наследник содержал один экземпляр полей класса-предка, необходимо использовать механизм виртуального наследования (virtual-наследование).

В ролевой политике управления доступом при задании отображения RP с учетом ролевой иерархии возможны два подхода: листовой, когда полномочия раздаются только листовым ролям, а далее распределяются между ролями по принципу наследования, и охватный, при котором набор полномочий старших ролей может пополняться и не унаследованными полномочиями. В объектной интерпретации эти два подхода регламентируются запретом или разрешением на пополнение производных классов-ролей дополнительными (не унаследованными) полями. В объектно-ориентированных языках программирования производный класс всегда может дополнить базовый – реализован более общий охватный принцип. Поэтому, в объектной модели ролевой политики безопасности необходимо предусмотреть спецификатор наследования, запрещающий расширение множества элементов производного класса. Назовем такое наследование «нерасширяемым» (non-expandable).

Стоит отметить, что, так как полномочия являются не методами, а полями классов-ролей, они не могут быть переопределены (изменить свое поведение) в производных классах за счет механизма виртуализации, что согласуется с требованиями безопасности.

Так как рассмотренные элементы составляют описательную часть модели, все поля и методы классов-полномочий, классов-ролей и классов-пользователей должны быть скрытыми. В силу наличия отношения наследования, целесообразно использовать спецификатор доступа protected.

Соответствие между основными элементами базовой ролевой модели управления доступом и объектной спецификацией представлены в таблице 1: основные множества заданы наборами классов, а основные отображения – отношениями между классами.

Таблица 1: Объектная спецификация основных элементов ролевой модели управления доступом

Ролевая модель	*ООП*
Полномочия	Классы P1, ..., Pm
Роли	Классы R1, ..., Rn
Пользователи	Классы U1, ..., Us
Назначение полномочий $p_{i_1}, \ldots, p_{i_u}$ роли r_i	Агрегация класса Ri (контейнер) и классов Pi1, ..., Piu (protected-поля)
Назначение ролей $r_{j_1}, \ldots, r_{j_v}$ пользователю u_j	Агрегация класса Uj (контейнер) и классов Rj1, ..., Rjv (protected-поля)
Ролевая иерархия	Множественное виртуальное public-наследование между классами R1, ..., Rn

2 Элементарные операторы администрирования ролевой модели

Стандарт ролевого управления доступом предполагает некоторый набор элементарных операторов (функций) для администрирования ролевой модели [2, 11]. Функции-операторы можно разделить на три категории: основные, вспомогательные и информационные. Эти функции легко могут быть выражены в терминах преобразования построенной системы классов. В таблице 2 представлены первые две категории.

Спецификация основных функций-операторов очевидным образом следует из объектной интерпретации основных множеств и отображений ролевой модели. Более детального обсуждения требуют операторы вспомогательной категории.

Основной задачей процесса управления доступом является задание отображения CU. Пусть пользователя u_j в текущем сеансе c необходимо авторизовать на роли $CU(c, u_j) \subseteq UR(u_j)$. В объектной модели этому процессу соответствует создание класса-наследника CUj базового класса Uj, в котором поля-роли, определяемые множеством ролей $CU(c, u_j)$, из скрытых перемещались бы в интерфейс класса. Это можно сделать, разместив в классе CUj объявления using Uj::Rx; для всех полей Rx, нуждающихся в перемещении. Очевидно, наследование должно быть «нерасширяемым».

Необходимо заметить, что в языках программирования объект производного класса одновременно является объектом любого из родительских классов. Тем самым будет выполнено одно из основных требований к построению отображения CU: вместе с заданной ролью пользователь должен быть авторизован и на все подчиненные роли.

Завершение текущего сеанса работы пользователя u_j может быть интерпретировано как уничтожение класса CUj. Так как уничтожаемый класс не имеет наследников, процедура преобразования иерархии не требует перемещения элементов между классами.

Таблица 2: Объектная интерпретация элементарных операторов администрирования ролевой политики управления доступом

Ролевая модель	*ООП*
Основные операторы	
Создать / удалить роль r_i.	Создать / удалить класс Ri (при удалении класса Ri провести преобразование классов U1, ..., Us).
Создать / удалить пользователя u_j.	Создать / удалить класс Uj.
Приписать / ликвидировать полномочие p_k роли r_i.	Добавить / удалить в классе Ri поле-ссылку на объект класса Pk.
Разрешить авторизацию / запретить авторизацию пользователя u_j на роль r_i.	Добавить / удалить в классе Uj поле-ссылку на объект класса Ri.
Создать / удалить отношение наследования между существующими ролями r_{i_1} и r_{i_2} ($r_{i_2} \geq r_{i_1}$).	Провести преобразование иерархии классов R1, ..., Rn с целью порождения наследования / свертывания иерархии для классов Ri1 (базовый класс) и Ri2 (класс-наследник).
Вспомогательные операторы	
Открыть сеанс работы пользователя u_j с активацией требуемого набора ролей (авторизовать пользователя).	Осуществить простое «нерасширяемое» public-наследование класса CUj от класса Uj с перемещением требуемых для авторизации полей в интерфейс класса CUj.
Завершить сеанс работы пользователя u_j.	Удалить класс CUj.
Проверить правомерность доступа – получить набор полномочий, доступных пользователю u_j по всем ролям, на которые он получил авторизацию в текущем сеансе.	Для каждого поля из интерфейса класса CUj получить набор его доступных полей.

Заметим, что за рамками рассмотрения остались ограничения на разделение обязанностей и вспомогательные функции. При необходимости они также могут быть представлены в терминах анализа и преобразования объектной ролевой модели.

3 Пример создания классов, описывающих заданную ролевую модель

Рассмотрим пример объектно-ориентированной спецификации некоторой ролевой политики управления доступом. Пусть $R = \{r_1, r_2, r_3, r_4, r_5\}$, $P = \{p_1, p_2, p_3, p_4\}$, $U = \{u_1, u_2, u_3\}$. Иерархия ролей представлена графом на рисунке 1. Пусть «листовым» ролям приписаны следующие наборы полномочий: $RP(r_1) = \{p_1, p_2\}$, $RP(r_2) = \{p_1, p_3\}$, $RP(r_3) = \{p_3, p_4\}$. Будем считать, что полномочия по иерархии распространяются согласно листовому подходу. Тогда $RP(r_4) = \{p_1, p_2, p_3\}$, $RP(r_5) = \{p_1, p_2, p_3, p_4\}$. Определены разрешенные для авторизации роли для каждого пользователя: $UR(u_1) = \{r_1, r_3\}$, $UR(u_2) = \{r_3, r_4\}$, $UR(u_3) = \{r_5\}$.

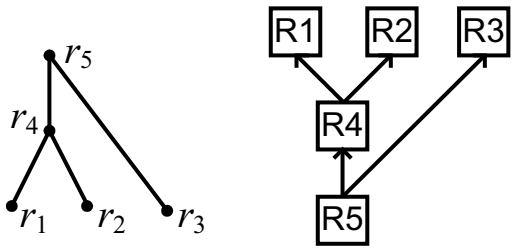

Рис. 1: Ролевая иерархия (слева) и соответствующая иерархия классов (справа)

Описание этой ролевой модели в виде классов, связанных отношениями наследования (см. рис. 1) и агрегации, будет иметь следующий вид:

```
class P1{};   class P2{};   class P3{};   class P4{};

 class R1{          class R2{          class R3{
  protected:         protected:         protected:
   P1 *p1;            P1 *p1;            P3 *p3;
   P2 *p2;            P3 *p3;            P4 *p4;
 };                 };                 };
class R4: virtual public non-expandable R1, virtual public non-expandable R2{};
class R5: virtual public non-expandable R4, virtual public non-expandable R3{};

 class U1{          class U2{          class U3{
  protected:         protected:         protected:
   R1 *r1;            R3 *r3;            R5 *r5;
   R3 *r3;            R4 *r4;          };
 };                 };
```

Рассмотрим вариант администрирования ролевой модели. Авторизация пользователя u_2 на роль r_4 в текущем сеансе работы системы будет заключаться в создании следующего класса:

```
class CU2: public non-expandable U2{
 public:
  using U2::r4;
};
```

Окончательный вариант UML-диаграммы классов, сопоставленных предложенной ролевой политике управления доступом, представлена на рисунке 2.

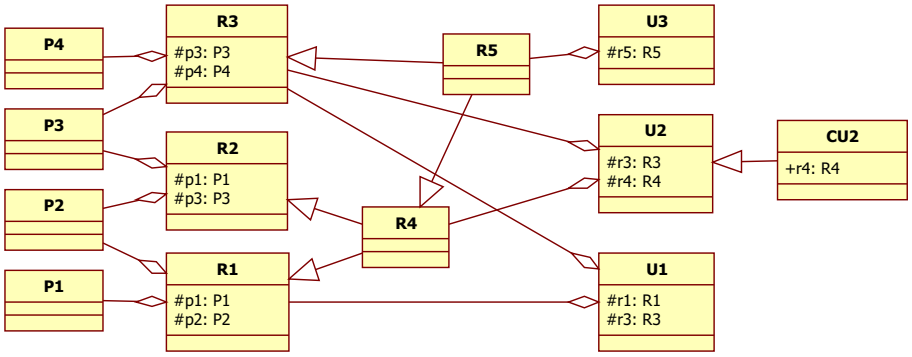

Рис. 2: UML-диаграмма классов, соответствующих заданной ролевой модели

Заключение

Подводя итог, еще раз отметим, что объектная спецификация ролевой модели управления доступом проводится в терминах классов и отношений между ними: агрегации и наследования. Непосредственно управление доступом (функционирование) описывается с позиций метапрограммирования. При таком подходе стирается разница между данными и программой. Так авторизация пользователя в текущем сеансе работы системы интерпретируется как создание класса, находящегося в некотором отношении с классами существующей структуры.

Объектно-ориентированное расширение ролевой модели позволит обосновать применимость теоретических и практических результатов, полученных для ролевых политик управления доступом в компьютерных системах микроуровня, к БИС, для которых построение эффекивной подсистемы защиты информации требует особых подходов в связи с возникновением дополнительных задач в управлении доступом [12].

Список литературы

[1] W. Essmayr, G. Pernul, A. M. Tjoa. Access Controls by Object-Oriented Concepts. *Database Security XI / Part of the Series IFIP Advances in Information and Communication Technology*, 325–340, 1998.

[2] V. A. Galatenko. *Osnovy Informatsionnoi Bezopasnosti* M., Internet-Universitet Informatsionnykh Ttekhnologii – INTUIT.ru, 2003 (In Russian).

[3] URL: `http://www.object-oriented-security.org/`.

[4] *Role Based Access Control (RBAC) and Role Based Security.* URL: `http://csrc.nist.gov/groups/SNS/rbac/`.

[5] J. Barkley. Implementing Role Based Access Control Using Object Technology. *First ACM Workshop on Role-Based Access Control*, 1995. URL: `http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=916544`.

[6] P. P. Oleinik, S. M. Salibekian. Model of Security for Object-Oriented and Object-Attributed Applications. *Trudy ISP RAN*, 28(3):35–50, 2016 (In Russian).

[7] S. V. Belim , S. Yu. Belim, S. V. Usov. The Object-Oriented Modificaction of Models of Security HRU. *Problemy Informatsionnoi Bezopasnosti. Komp'iuternye Sistemy*, 1:6–14, 2010 (In Russian).

[8] S. V. Usov. On the Relation Between the Object-Oriented Discretionary Security Model and The Subject-Object Mandatory Model. *Mathematical Structures and Modeling*, 4(40):151–163, 2016 (In Russian).

[9] URL: `https://habrahabr.ru/post/227753/`.

[10] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2): 38-47, 1996.

[11] S. V. Belim, N. F. Bogachenko, I. A. Firdman. Investigation of the Permissions Seepage in a Role-Based Access Control. *Problemy Informatsionnoi Bezopasnosti. Komp'iuternye Sistemy*, 3:7–13, 2012 (In Russian).

[12] A. A. Volodina, I. M. Levkin. Adaptivnyi Podkhod k Zashchite Informatsii v Bol'shikh Informatsionnykh Sistemakh. *Problema Kompleksnogo Obespecheniia Informatsionnoi Bezopasnosti i Sovershenstvovanie Obrazovatel'nykh Tekhnologii Podgotovki Spetsialistov Silovykh Struktur.* Mezhvuzovskii Sbornik Trudov VI Vserossiiskoi Nauchno-Tekhnicheskoi Konferentsii KONFIB'15. SPb, Universitet ITMO, 65–73, 2016 (In Russian).

Role-Based Access Control Metamodel

Nadezda F. Bogachenko

An object-oriented role-based access control metamodel is suggested in this article. The basic sets and mappings are described in the terms of classes and relations of inheritance and aggregation. An access control is considered from the metaprogramming point of view.