УДК 681.3.053

ОБНАРУЖЕНИЕ И УСТРАНЕНИЕ ПРОТИВОРЕЧИЙ В СПЕЦИФИКАЦИЯХ СЛОЖНЫХ СИСТЕМ[1]

А.В. Тишков, О.В. Черватюк,
Д.П. Лакомов, С.А. Резник, Е.В. Сидельникова[2]

В работе описывается подход к верификации спецификаций сложных систем на примере политик безопасности компьютерных сетей. Особенностью подхода является применение гибридной архитектуры, использующей разные математические подходы для поиска и разрешения различных конфликтов. Представлены модели реализации модуля верификации с применением исчисления событий и абдуктивного вывода, и модуля, использующего технологию верификации на модели.

Введение

Системы управления, основанные на политиках, получают все большее распространение в самых разных сферах применения благодаря гибкости в управлении и удобству администрирования. Политики, как совокупности правил функционирования системы в виде “если-то”, встречаются в настройках различных приложений — от небольших наборов правил распределения писем почтовой программы, формируемых одним пользователем, до сложных комплексов правил реализации бизнес-процессов в сложных распределенных ERP-приложениях, создаваемых и поддерживаемых коллективом разработчиков и администраторов.

Чем сложнее система, основанная на политиках, тем актуальней становится задача поддержки непротиворечивости правил политики.

[1] Работа выполнена при финансовой поддержке РФФИ (проект №04-01-00167), программы фундаментальных исследований ОИТВС РАН (контракт №3.2/03) и при частичной поддержке, осуществляемой в рамках проекта POSITIF Шестой рамочной программы Евросоюза (контракт IST-2002-002314). Авторы благодарят И.В. Котенко за помощь в подготовке статьи.

[2] *199178, С.-Петербург, 14 линия, 39, СПИИРАН, в порядке перечисления авторов: avt@iias.spb.su, ovch@computer.edu.ru, dmll@computer.edu.ru, sergeyreznick@yandex.ru, KittyKate137@yandex.ru*

Создавая множество правил различных типов и категорий, пользователь системы не в состоянии без соответствующего инструментария отследить возможные конфликты между правилами политики, особенно в ситуации взаимного влияния правил разных категорий. Другим не менее важным условием использования создаваемой политики является соответствие возможностей используемых программных и технических средств требованиям политики. Полная и непротиворечивая политика может стать бессмысленной, если в системе есть хотя бы одно устройство, не способное реализовать возложенные на него функции, сформулированные в правилах.

Таким образом, актуальной задачей для современных систем управления является задача поддержания непротиворечивости в спецификациях их поведения, т.е. обнаружения и разрешения возможных конфликтов. Данная задача давно привлекает внимание специалистов в области искусственного интеллекта, в первую очередь, как задача верификации баз знаний сложных интеллектуальных систем [Ayel et al., 1991; Tsai et al., 2000; Muller et al., 2000].

В настоящей работе описывается подход к верификации спецификаций сложных систем на примере политик безопасности компьютерных сетей. Особенностью подхода является применение гибридной архитектуры, использующей разные математические подходы для поиска и разрешения различных типов конфликтов, открытость для введения дополнительных моделей и методов верификации, а также использование человеко-машинных процедур разрешения противоречий. Программная система верификации, реализующая данный подход, является своеобразным “отладчиком” политики, предоставляя администратору инструмент обнаружения и разрешения конфликтов между правилами политики. Система ориентирована на разрешение конфликтов как в рамках одной категории (аутентификации, авторизации, фильтрации, конфиденциальности передачи информации и данных, операционных правил), так и разных категорий, а также обнаружения невозможности реализации правил политики при использовании заданной конфигурации компьютерной сети и определения стратегии для устранения этого конфликта.

Исследуя опубликованные работы по обнаружению и разрешению конфликтов в политиках безопасности, мы столкнулись с определенными трудностями в их использовании. Часть работ достаточно четко задает понятие конфликта, но эти конфликты определяются лишь для одной из категорий. Так, в работах [Jajodia et al., 1997; Jajodia et al., 2001; Vimercati et al., 2003; Zhang et al., 2005] исследуется конфликт авторизации. В работах [Lupu et al., 1997; Lupu et al., 1999; Lymberopoulos et al., 2004] определяется несколько типов конфликтов, которые связаны с

авторизацией и обязательным выполнением, близким к понятию операционного правила. Изложенные в [Dunlop et al., 2003; Dunlop et al., 2002] решения, основанные на темпоральной логике, требуют дополнительных понятий, таких как состояние системы и активное, пассивное и потенциальное состояние конфликта. Другие работы, определяющие понятие конфликта, очень сложно или даже невозможно вырвать из контекста вводимого в них языка или системы. Так, интересный подход, связанный с деонтической логикой [Cholvy et al., 1995; Cholvy et al., 1997; CIM, 2006] требует достаточно специфического определения роли и ввода иерархии ролей.

Дальнейшее изложение структурировано следующим образом. Во первом разделе рассмотрена архитектура предлагаемой системы верификации политик безопасности. Во втором и третьем разделах представлены модели реализации двух базовых модулей верификации: (1) основанном на теории доказательств, с применением исчисления событий и абдуктивного вывода, и (2) использующем технологию верификации на модели (model checking). В четвертом разделе кратко описана текущая реализация прототипа системы верификации.

1. Архитектура системы верификации политик безопасности

Архитектура системы верификации политик безопасности, реализующей предлагаемый подход, является многомодульной. Специально выделенный менеджер верификации запускает модули верификации для проверки непротиворечивости политики и собирает от них информацию о найденных конфликтах и способах их разрешения. Модули верификации осуществляют эту проверку на основе различных математических методов. Ядро системы верификации политик безопасности содержит два основных класса: VerificationManager и VerificationModule.

Основными входными данными для системы верификации являются описание политики и системы. Для описания системы и политик используется разработанный язык, представленный набором XML-схем и основанный на стандарте CIM. Модули верификации транслируют XML-представление политик и системы в представление, необходимое для технологии, используемой каждым из модулей при верификации.

Выходными данными работы системы являются: информация об обнаруженном конфликте, включающая категорию конфликта; элементы компьютерной системы, корректная работа которых будет под угрозой в результате этого конфликта; правила, участвующие в конфликте; возможные стратегии разрешения.

2. Модуль верификации, основанный на исчислении событий

Исчисление событий является многосортной теорией предикатов первого порядка, впервые введенной в [Kowalski et al., 1986]. В основу исчисления событий заложен “принцип инерции” — свойства окружающего мира изменяются только под воздействием событий и остаются неизменными в промежутках между ними. Вводятся два основных сорта: свойство (fluent), представляющее некоторое изменяемое во времени состояние системы, и событие (event). Чтобы описать простейшее исчисление событий требуются следующие семь предикатов: InitiallyTrue(*f*) — свойство *f* выполняется в начальный момент времени; InitiallyFalse(*f*) — свойство *f* не выполняется в начальный момент времени; Happens(*e,t*) — событие *e* происходит в момент времени *t*; Initiates(*e,f,t*) — если *e* произошло в момент времени *t*, оно инициировало свойство *f*; Terminates(*e,f,t*) — если *e* произошло в момент времени *t*, оно терминировало свойство *f*; HoldsAt(*f,t*) — свойство *f* выполняется в момент времени *t*; Clipped(t_1,*f*,t_2) — свойство *f* было терминировано промежутке между моментами t_1 и t_2.

Для применения абдуктивного вывода [Kakas et al., 1993; Endriss et al., 2004] предметно-независимую аксиоматику удобно определять в форме тождеств:

- HoldsAt(*f,t*) ≡ [Happens(*e*,t_1) Λ Initiates(*e*,*f*,t_1) Λ $t_1<t$ Λ ¬(Clipped(t_1,*f,t*))] V [InitiallyTrue(*f*) Λ ¬(Clipped(0,*F,T*))] V [*t*=0 Λ InitiallyTrue(*f*)];
- InitiallyTrue(*f*) ≡ ¬(InitiallyFalse(*f*));
- Clipped(t_1,*f*,t_2) ≡ Happens(*e,t*) Λ $t_1<t$ Λ $t<t_2$ Λ Terminates(*e*,*f*,*t*).

Для конкретной задачи необходимо ввести предметную аксиоматику, определяющую события, которые инициируют соответствующие свойства. Для примера рассмотрим события, относящиеся к политикам авторизации или аутентификации, инициируют (либо останавливают) соответствующие свойства. Таким образом, получаем следующую предметную аксиоматику:

- Initiates(RequestAuthentication ((?user, AuthenticationMethod , Target), Authenticated (?user, AuthenticationMethod, Target), ?*t*);
- [HoldsAt(Authenticated (?user, AuthenticationMethod, Target), ?*t*)] Λ [SubjectRole (?user, PermittingRole)] → Initiates(RequestAuthorization (?user, Activity, Target), Authorized (?user, Activity, Target), ?*t*);
- [HoldsAt(Authenticated (?user , AuthenticationMethod , Target), ?*t*)] Λ [SubjectRole (?user, PermittingRole)] → Initiates(RequestAuthorization (?user, Activity, Target), AllowAuthorization (?user, Activity, Target), ?t);

- [HoldsAt(Authenticated (?user, AuthenticationMethod, Target), ?*t*)] Λ [SubjectRole (?user, ForbiddingRole)] → Initiates(RequestAuthorization ((?user,Activity,Target), DenyAuthorization (?user,Activity, Target), ?t).

Рассмотрим пример (табл. 1), в котором задаются правила аутентификации на один и тот же объект (SMTP Server), требующие различного типа аутентификации для одного субъекта (пользователя, принадлежащего роли Administrator).

Табл. 1.

№	Объект	Действие	Субъект	Тип аутентификации	Тип политики
1	SMTP Server	Configure	Administrator	Account	Аутентификация, Авторизация
2	SMTP Server	Send, Relay	Administrator, Student, ...	Shared secret	Аутентификация, Авторизация

Конфликт возникает, когда пользователь пытается аутентифицироваться при помощи обоих типов аутентификации. Конфликтующие условия пользователь-объект можно задать следующим предикатом:

AuthenticationConflict (?user, ?target) ≡ HoldsAt (Authenticated (?user, ?authType$_1$, ?target), ?*t*) Λ HoldsAt(Authenticated (?user, ?authType$_2$, ?target), ?t) Λ (?authType$_1$!= ?authType$_2$).

Это выражение будет использовано в качестве запроса для абдуктивного вывода.

Для обнаружения конфликта используется абдуктивный вывод. В общем случае, абдуктивная логическая программа представляет собой тройку (Th, IC, *Q*), в которую входят теория Th, конечное множество ограничений целостности IC и запрос *Q*. Теорией называется множество так называемых iff-определений:

$$p(X_1, \ldots, X_k) \equiv D_1 \vee \ldots \vee D_k$$

Предикат *p* не может быть специальным предикатом (сравнения, =, TRUE и FALSE). Каждый из дизъюнктов D_i является конъюнкцией литералов. Предикат *p* называется определенным. Предикат, который не является ни определенным, ни специальным называется абдуктивно-выводимым. Ограничения целостности, из которых состоит множество IC, являются импликациями следующего вида:

$$L_1 \wedge \ldots \wedge L_m \rightarrow A_1 \vee \ldots \vee A_n,$$

где для любого *i* L_i является литералом, а A_i атомом.

Запрос *Q* — это конъюнкция литералов. Используется следующее определение абдуктивного метода: ответом на запрос *Q* соответствующей абдуктивной программы (Th,IC,*Q*) называется пара (Δ,σ), где Δ – конечное

множество абдуктивно-выводимых атомов, а σ –подстановка свободных переменных, встречающихся Q, такие что

$$\text{Th } V \text{ Comp}(\Delta) \models \text{IC} \wedge Q_\sigma.$$

4. Модуль верификации на модели

Методы проверки на модели основаны на переборе состояний, в которые может перейти система в зависимости от запросов пользователей и ответов компонента, принимающего решения о разрешении или отклонении такого запроса. Перебор управляется условиями, которые сформулированы на языке темпоральной логики и выражают корректные состояния системы. Модуль верификации на модели строит модель, описывающую поведение защищаемой системы на языке Promela, в которой конфликт определяется на языке темпоральной логики как нежелательное состояние системы.

SpinVerificationModule реализован как Java-класс, способный разбирать правила, порождать соответствующие Promela-описания, подавать их на вход SPIN [Holzmann et al., 1997] для порождения верификатора, компилировать верификатор, запускать его и анализировать его выход с тем, чтобы определить результат верификации. В случае конфликта, обнаруженного путем верификации, порожденная SPIN трасса анализируется с тем, чтобы определить конфликтующие правила. В случае отсутствия конфликтов, сообщается об успешной верификации.

5. Реализация прототипа системы верификации

Разработан программный прототип системы верификации. Ядро системы представляет собой Java-библиотеку, на основе которой создано web-приложение и приложение, запускаемое на отдельной рабочей станции. Оба приложения поддерживают следующую функциональность: регистрация внешнего модуля верификации; выбор политики безопасности для верификации; загрузка и запуск зарегистрированных модулей верификации; отчет о результатах верификации, содержащий перечень конфликтов (если найдены) и перечень стратегий разрешения для каждого конфликта; разрешение конфликта при помощи одной из стратегий разрешения; просмотр журнала работы модуля верификации, включающего внутреннее представление политики в соответствии с формализмом, применяемым конкретным модулем, и результаты верификации.

На рис. 1 представлены основные окна интерфейса приложения для отдельной рабочей станции. В левой части рис. 1 изображены следующие окна: основное окно системы (вверху), окно результата верификации

(посередине), окно выбора стратегии разрешения конфликта (внизу справа) и окно реализации одной из стратегий разрешения — деактивации конфликтующих правил вручную (внизу слева). В правой части рисунка изображены журналы работы модулей верификации: для исчисления событий (вверху) и проверки на модели (внизу). В нижней части каждого из журналов выделены отчеты о результатах верификации.

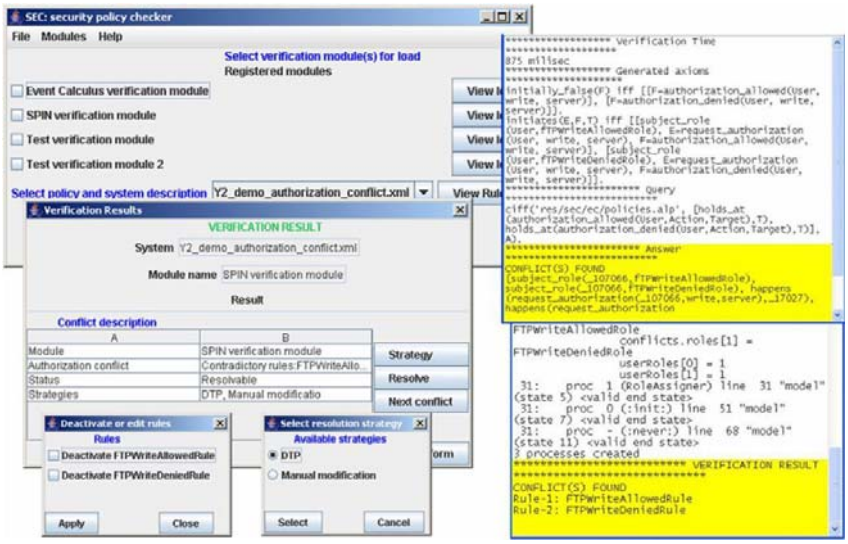

Рис. 1. Основные окна интерфейса системы верификации

6. Заключение

В настоящей статье на примере системы верификации политик безопасности компьютерных сетей представлен общий подход к верификации спецификаций сложных систем. Предложена архитектура системы верификации и рассмотрены два подхода к реализации механизмов обнаружения и разрешения противоречий в политиках безопасности. Прототип системы реализован на языке Java с использованием SICStus Prolog 3.12.5, CIFF 3.0, SPIN 4.2.6. В настоящее время ведутся работы по формализации конфликтов в правилах пяти категорий безопасности. В дополнение к аутентификации и авторизации, затронутым в данной работе, рассматриваются правила фильтрации сетевого трафика, конфиденциальности данных и операционные правила.

Список литературы

[Ayel et al., 1991] Ayel M., Laurent J.P. (eds.) Validation, Verification, and Test of Knowledge-Based Systems, John Wiley and Sons, Chichester, U.K., 1991.

[Cholvy et al., 1995] Cholvy L., Cuppens F. Solving normative conflicts by merging roles // Proceedings of the fifth International Conference on Artificial Intelligence and Law. Washington. May 1995.

[Cholvy et al., 1997] Cholvy L., Cuppens F. Analysing consistency of security policies // Proceedings of IEEE Symposium on Security and Privacy. Oakland, USA. May 1997.

[CIM, 2006] Common Information Model (CIM) Standards. http://www.dmtf.org/standards/cim . 2006.

[Dunlop et al., 2003] Dunlop N., Indulska J., Raymond K. Methods for Conflict Resolution in Policy-Based Management Systems // EDOC. 2003. P. 98-111.

[Dunlop et al., 2002] Dunlop N., Indulska J., Raymond K. Dynamic Conflict Detection in Policy-Based Management Systems // EDOC. 2002. P. 15-26.

[Endriss et al., 2004] Endriss U., Mancarella P., Sadri F., Terreni G., Toni F. The CIFF Proof Procedure: Definition and Soundness Results // Technical Report 2004/2, Imperial College London, May 2004.

[Holzmann et al., 1997] Holzmann G. The Spin Model Checker // IEEE Trans. on Software Engineering, May 1997. Vol.23 № 5. P.279-295.

[Jajodia et al., 1997] Jajodia S., Samarati P., Subrahmanian V. S. A Logical Language for Expressing Authorizations // IEEE Symposium on Security and Privacy. 1997.

[Jajodia et al., 2001] Jajodia S., Samarati P., Sapino M. L., Subrahmanian V. S. Flexible support for multiple access control policies // ACM Trans. Database Systems. 2001. Vol. 26, № 2. P. 214-260.

[Kakas et al., 1993] Kakas A.C., Kowalski R.A., Toni F. Abductive Logic Programming // Journal of Logic and Computation. 1993. Vol 2, № 6, P. 719-770.

[Kowalski et al., 1986] Kowalski R.A, Sergot M.J. A Logic-Based Calculus of Events // New Generation Computing. 1986. № 4. P. 67–95.

[Lupu et al., 1997] Lupu E., Sloman M. Conflict Analysis for Management Policies // Fifth IFIP/IEEE International Symposium on Integrated Network Management (IM'97), San-Diego, 1997. P. 430-443.

[Lupu et al., 1999] Lupu E., Sloman M. Conflicts in Policy-based Distributed Systems Management // IEEE Transactions on Software Engineering. 1999. Vol 25, № 6. P. 852-869.

[Lymberopoulos et al., 2004] Lymberopoulos L., Lupu E., Sloman M. Ponder Policy Implementation and Validation in a CIM and Differentiated Services Framework // IFIP/IEEE Network Operations and Management Symposium (NOMS 2004). Seoul, Korea. April 2004.

[Muller et al., 2000] Muller J.H., Dieng R. (eds.). Computational Conflicts. Conflict Modeling for Distributed Intelligent Systems. Springer. 2000.

[Tsai et al., 2000] Tsai W.-T., Vishnuvajjala R., Zhang D. Verification and Validation of Knowledge-Based Systems // IEEE Transactions on Knowledge and Data Engineering, Vol.11, No. 1, 1999.

[Vimercati et al., 2003] De Capitani di Vimercati S., Paraboschi S., Samarati P. Access control: principles and solutions // Software – Practice and Experience. 2003. Vol. 33, № 5. P. 397-421.

[Zhang et al., 2005] Zhang N., Ryan M.D., Guelev D. Evaluating Access Control Policies Through Model Checking // Lecture Notes in Computer Science. Vol. 3650, P. 446-460, Springer-Verlag, 2005.