یک مدل کنترل دسترسی انعطاف‌پذیر برای پایگاههای دادهٔ رابطه‌ای

امید خوانساری‌نیا و رسول جلیلی
دانشکدهٔ مهندسی کامپیوتر، دانشگاه صنعتی شریف
{khansary.ce, jalili}.sharif.edu

چکیده

در این مقاله یک مدل کنترل دسترسی ارائه شده است که می‌تواند برای اِعمال همزمان تعدادی از سیاستهای امنیتی سطح بالا در پایگاههای دادهٔ رابطه‌ای به کار رود. در این مدل، امکان استفادهٔ توأم از مجوزهای مثبت ومنفی و نیز استثناءپذیری مجوزهای اعطا شده توسط کاربران وجود دارد. همچنین از مدیریت مجوزها در هر سطحی از سلسله‌مراتب گروه همراه با تعلیق موقتی آنها پشتیبانی می‌شود. مدیریت این مدل، ترکیبی از دو سیاست مالکیت و غیرمتمرکز است. مجوزهای مدیریتی می‌تواند به گونه‌ای واگذار گردد که کاربران بتوانند کنترل مجوزها را روی جداول تحت مالکیت خود حفظ کنند.

کلمات کلیدی

کنترل دسترسی، گراف کنترل دسترسی، سلسله‌مراتب گروه، مجوز، مدیریت، واگذاری، سازگاری و پایگاه دادهٔ رابطه‌ای.

۱- مقدمه

هدف از کنترل دسترسی، محدود کردن اعمال و فعالیتهای یک کاربر معتبر سیستم کامپیوتری و برنامه‌های اجرایی وی می‌باشد [۱]. سیاستهای امنیتی مورد نظر کاربران مختلف برای دسترسی به داده‌های گوناگون، متفاوت است: داده‌هایی که همهٔ کاربران می‌توانند به آنها دسترسی داشته باشند، داده‌هایی که فقط کاربران خاصی باید به آنها دسترسی یابند، و داده‌هایی که تمامی کاربران به جز تعدادی خاص می‌توانند به آنها دسترسی داشته باشند. بنابراین نیاز به یک مدل کنترل دسترسی انعطاف‌پذیر و قدرتمند که بتواند خواسته‌های یک کاربر را با کمک سیاستهای سطح بالا برآورده سازد، به شدت احساس می‌شود. برخی از این سیاستها عبارتند از [۲]:

- *پشتیبانی از مجوزهای مثبت و منفی.* مجوز مثبت بیانگر دسترسیهای مجاز و مجوز منفی معرف دسترسیهای نفی شده است. اکثر مدلهای موجود با اِعمال سیاستهای باز یا بسته، فقط از یک نوع مجوز پشتیبانی می‌کنند. پشتیبانی از هر دو نوع مجوز به کاربر اجازه می‌دهد تا دقیقاً آنچه را که مدّ نظر اوست، بر اساس وضعیت و شرایط موجود بیان کند.
- *پشتیبانی از استثنائات و مجوزهای قوی.* گاهی اوقات لازم است که یک مجوز به اکثریت افراد به جز چند نفر خاص اعطا شود. برای مثال فرض کنید که همهٔ کاربران به جز «علی» مجاز به خواندن داده‌ای باشند. در این حالت بهتر است

که مجوز عمل خواندن، به گروه شامل تمامی کاربران اعطا شده و برای کاربر «علی» استثناء بیان گردد (با کمک مجوز منفی). معمولاً مطلوب است که پذیرش استثناء برای هر مجوزی که توسط کاربران اعطا می‌شود، با تعیین قوی یا ضعیف بودن آن مشخص گردد. در اصطلاح، مجوزهایی که استثناء نمی‌پذیرند، قوی و مجوزهایی که می‌توانند استثناء داشته باشند، ضعیف نامیده می‌شوند.

- *مدیریت*. هر مدل کنترل دسترسی به منظور اعطای (لغو) مجوزها به (از) دیگران به یک سیاست مدیریتی نیاز دارد. رایج‌ترین سیاستهای مدیریتی عبارتند از: سیاست متمرکز (که در آن یک کاربر یا گروهی از کاربران، مجوزها را تعیین می‌کنند)، سیاست مالکیت (که در آن هر کاربر، مدیریت دسترسی به اشیائی که خود آنها را ایجاد کرده به عهده دارد) و سیاست غیرمتمرکز (که در آن کاربر می‌تواند مجوزهای اعطا و لغو مجوز برای اشیاء تحت مالکیت خود را به کاربران دیگر نیز واگذار کند).
- *امکان واگذاری*[1] *و حفظ کنترل*. هرچند سیاستهای غیرمتمرکز امکان واگذاری مجوزها را به وجود می‌آورند ولی این اشکال را نیز دارند که مالک یک شئ ممکن است کنترل مجوزهای آن را از دست بدهد. مثلاً در صورت پشتیبانی از استثناء، مجوزهای تعیین شده توسط مالک می‌تواند به وسیلهٔ کاربران دیگر همپوشانی شود. بنابراین بهتر است مجوزهای مدیریتی به شکل محدود شده‌ای واگذار گردد.
- *دسته‌بندی عاملها*. این دسته‌بندی به وسیلهٔ گروه یا نقش انجام می‌گیرد. گروه، مجموعه‌ای از کاربران و نقش، مجموعه‌ای از مجوزهاست [۳]. اغلب، مجوزها برای دسته‌ای از کاربران که دارای ویژگیهای مشترکی هستند یکسان می‌باشند (مانند کارمندان شاغل در یک بخش اداره). بنابراین در صورت عدم وجود این سیاست، مدیر امنیت سیستم با تغییر عضویت هر کاربر، باید مجوزهای زیادی را لغو و اعطا کند که امری پیچیده و خسته‌کننده است.

هدف از این مقاله، ارائهٔ یک مدل کنترل دسترسی برای پایگاههای دادهٔ رابطه‌ای همراه با اِعمال سیاستهای امنیتی فوق است. در این مدل، امکان استفاده از مجوزهای مثبت و منفی که هر یک می‌توانند قوی یا ضعیف باشند وجود دارد. دسته‌بندی عاملها با کمک مفهوم سلسله‌مراتب گروه پشتیبانی می‌شود و استثناء براساس این سلسله‌مراتب و در هر سطحی قابل اعمال است. مجوزهای منفی به صورت مجوزهای بلوکه‌کننده پشتیبانی می‌شوند، بدین معنی که اگر به کاربری مجوز منفی داده شود که تقدم بر مجوز مثبت دارد، این مجوز مثبت در مجموعهٔ مجوزها باقی می‌ماند هر چند که دیگر معتبر نیست و پس از لغو مجوز منفی دوباره معتبر خواهد شد. مدیریت با تلفیقی از دو سیاست مالکیت و غیرمتمرکز انجام می‌شود و امکان واگذاری برای هر مجوز وجود دارد. برای حفظ کنترل نیز مالک یک جدول، فقط باید مجوزهای ضعیف را واگذار کند.

ادامهٔ مقاله به شکل زیر سازمان یافته است. در بخش ۲ مروری بر کارهای انجام شده آمده است. بخش ۳ تعاریف و نمادهای لازم را معرفی می‌کند. در بخش ۴ مدل کنترل دسترسی و الگوریتمهای مربوطه ارائه می‌شود. بخش ۵ مدیریت مجوزها و بخش ۶ اطمینان از سازگاری حالت کنترل دسترسی سیستم را همراه با نحوهٔ رفع تصادم مابین مجوزهای ضعیف شرح می‌دهد. در بخش ۷ نیز نتیجه‌گیری آمده است.

۲- مروری بر کارهای انجام شده

هر چند مدلهای کنترل دسترسی زیادی تا به امروز توسط محققین ارائه گردیده است، ولی تعداد کمی از آنها از هر دو نوع مجوز مثبت و منفی پشتیبانی می‌کنند. در این بخش، فقط مدلهای معروفی که هر دو نوع مجوز را دارند مرور می‌شوند زیرا مدل

---

[1] delegation

ارائه شده در این مقاله نیز در این دسته قرار می‌گیرد. اولین مدل معروفی که با پشتیبانی از هر دو نوع مجوز، معرفی گردید مدل Andrew است که در سال ۱۹۸۹ برای حفاظت فایل در سیستم عامل ارائه شد [۴]. در همین سال یک مدل کنترل دسترسی انعطاف پذیر به نام Sea-View[۲] در دانشگاه استانفورد طراحی شد که در آن پشتیبانی از مجوز منفی توسط عملی به نام null انجام می‌شود [۵]. تصادم در هر دو مدل ذکر شده بر اساس تقدم نفی[۳] رفع می‌گردد.

قوی یا ضعیف بودن مجوزها، اولین بار در پروژه‌ای به نام ORION/ITASCA با هدف اِعمال کنترل دسترسی در سیستمهای شئ‌گرا در سال ۱۹۹۱ ارائه گردید [۶]. این مدل از نظر پذیرش سیاستهای امنیتی، نزدیکترین کار به مدل ارائه شده در این مقاله است ولی چند ویژگی متفاوت دارد که عبارتند از: تعریف مجوز برای نقش (و نه کاربر)، انتشار مجوز منفی از زیرنقش به نقش (و نه بالعکس) و جلوگیری از بروز تصادم بین مجوزهای ضعیف (و عدم وجود الگوریتمهای رفع تصادم).

مدل Bruggemann در سال ۱۹۹۲ بر اساس مفاهیم مدل ORION برای پایگاههای دادهٔ شئ‌گرا ارائه گردید که در آن تصادم بین مجوزها توسط تقدم اعداد نسبت داده شده به هر یک رفع می‌شود [۷]. مدل Shen و Dewan نیز در سال ۱۹۹۲ برای حفاظت اطلاعات در سیستمهای همکار معرفی شد که رفع تصادم بر پایهٔ تقدمهای ذکر شده در لیستهای کنترل دسترسی به شئ انجام می‌گیرد [۸]. مدل Gal-Oz نیز در سال ۱۹۹۳ برای سیستمهای شئ‌گرا معرفی شد که در آن مجوزهای یک رده توسط زیررده‌های آن به ارث برده می‌شود [۹].

اولین مدل کنترل دسترسی برای پایگاههای دادهٔ رابطه‌ای با پشتیبانی از هر دو نوع مجوز مثبت و منفی در سال ۱۹۹۷ توسط Bertino و همکاران وی، با توسعهٔ سیستم R [۱۰] با هدف حمایت از مجوزهای منفی ارائه گردید [۱۱]. این مدل در سال ۱۹۹۹ با ذکر الگوریتمهایی برای رفع خودکار تصادم بین مجوزها کاملتر شد [۲]. در جدول ۱ مقایسه‌ای بین این مدلها و مدل ارائه شده در این مقاله، از نظر پذیرش سیاستهای امنیتی مطرح شده در بخش ۱ انجام گرفته است.

جدول ۱- مقایسهٔ مدلهای کنترل دسترسی از نظر سیاستهای امنیتی مطرح شده در بخش ۱

مدل	سیاستهای امنیتی				
	مجوزهای مثبت و منفی	استثناء / اعمال مجوز قوی	مدیریت	واگذاری / حفظ کنترل	دسته‌بندی عاملها
Andrew	بلی	تقریباً / خیر	بلی	خیر / بی‌معنی	گروه
Sea-View	تقریباً	بلی / خیر	بلی	بلی / خیر	گروه
Orion	بلی	بلی / بلی	بلی	بلی / خیر	نقش
Bruggemann	بلی	بلی / بلی	خیر	بی‌معنی / بی‌معنی	نقش
Shen و Dewan	بلی	بلی / تقریباً	خیر	بی‌معنی / بی‌معنی	نقش
Gal-Oz	بلی	تقریباً / خیر	خیر	بی‌معنی / بی‌معنی	خیر
Bertino	بلی	تقریباً / خیر	خیر	بی‌معنی / بی‌معنی	گروه
جدید	بلی	بلی / بلی	بلی	بلی / بلی	گروه

۳- تعاریف و نمادها

تعریف شئ ، عامل و گراف عضویت. اشیای این مدل، جداول پایگاه داده هستند که به دو دستهٔ جداول حقیقی و مجازی تقسیم می‌شوند. جدول مجازی، همان دید[۴] است که توسط پرس‌وجوها با ارجاع به جداول حقیقی و مجازی دیگر تعریف

[۲] SEcure dAta VIEW

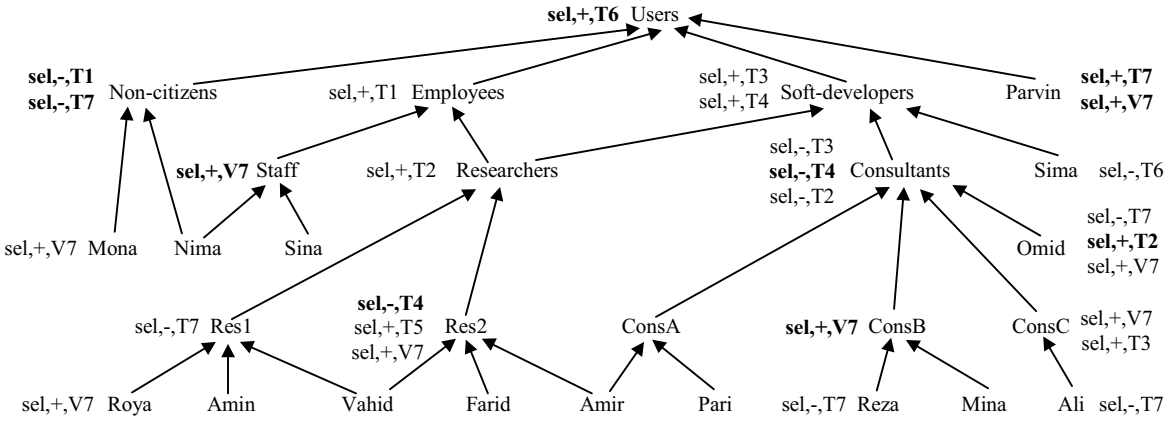
[۳] denial-takes-precedence

[۴] view

می‌شود. نماد $v \hookleftarrow t$ بیان می‌کند که در تعریف دید v، به جدول t (به صورت مستقیم یا غیرمستقیم) ارجاع داده شده است. مجموعهٔ جداول حقیقی با BT، مجموعهٔ جداول مجازی با VT و مجموعهٔ همهٔ جداول با $T = BT \cup VT$ نمایش داده می‌شود. فقط زیرمجموعه‌ای از کاربران مجاز، تحت نام مدیران پایگاه داده، قادر به ایجاد یا حذف جداول هستند. مجوزهای کاربران بر مبنای اَعمال قابل اجرا روی جداول، مانند select، update، insert و create-view تعیین می‌شوند.

عاملها، کاربران و گروهها هستند. ایجاد، حذف و اصلاح گروهها فقط به وسیلهٔ کاربران مجاز می‌تواند انجام شود. اعضای یک گروه را کاربران و یا گروههای دیگر تشکیل می‌دهند. عضویت عامل s در گروه G_j می‌تواند به دو شکل مستقیم یا غیرمستقیم باشد. عضویت مستقیم بدین معنی است که s به عنوان عضوی از G_j تعریف شده و با نماد $s \in_1 G_j$ نمایش داده می‌شود. s عضو غیرمستقیم G_j است ($s \in_n G_j$ و $n > 1$) اگر دنبالهٔ عاملهای $<s_1,\ldots,s_{n+1}>$ با شرط $s_1 = s$، $s_{n+1} = G_j$ و $s_i \in_1 s_{i+1}$ $(1 \le i \le n)$ موجود باشد. دنبالهٔ $<s_1,\ldots,s_{n+1}>$ مسیر عضویت s در G_j است که با نماد $mp(s,G_j)$ نشان داده می‌شود. $\in_0$ به معنای تساوی است (برای هر عامل، $s \in_0 s$) و $<s>$ بیانگر مسیر عضویت s در خودش است.

گروهها می‌توانند اعضای مشترک داشته باشند با این محدودیت که هیچ گروهی نمی‌تواند عضو خودش باشد (مستقیم یا غیر مستقیم). در نتیجه، یک عامل می‌تواند از مسیرهای مختلف عضو یک گروه باشد که نماد $MP(s_i,s_j)$ بیانگر مجموعهٔ تمامی مسیرهای عضویت از عامل s_i به عامل s_j است. رابطهٔ عضویت بین عاملها را می‌توان توسط *گراف عضویت* [۲] نمایش داد که در آن گرهها بیانگر عاملها و وجود یک لبه از s_i به s_j معرف عضویت مستقیم s_i در s_j است ($s_i \in_1 s_j$). چون هیچ عاملی عضو خودش نمی‌باشد گراف عضویت، یک گراف غیرحلقوی جهت‌دار است. مثالی از این گراف در شکل ۱ دیده می‌شود (بدون در نظرگیری مجوزهای اعطا شده به هر عامل).



شکل ۱- گراف حالت کنترل دسترسی

تعریف مجوز. برای نمایش مجوزهای مثبت و منفی از دو علامت + و - استفاده می‌شود. از آن جا که به کارگیری مجوز منفی بر روی دیدها، کنترل دسترسی را در عمل بسیار پیچیده و مشکل می‌کند در این مدل مجوز منفی فقط بر روی جداول حقیقی اعمال می‌شود. در صورت اعمال مجوز منفی روی دید v، هنگام دسترسی کاربری به v و برای جلوگیری از دسترسی به داده‌های نفی شده، باید تمامی مجوزهای دیدهایی که به طور مستقیم یا غیر مستقیم دارای جداول ارجاعی مشترک با دید v هستند بررسی گردد. البته در بیشتر موارد، زمانی که به یک مجوز منفی برای دید v نیاز باشد، می‌توان یک دید جدید که از نظر مفهوم متضاد با دید v است تعریف کرده و به آن مجوز مثبت داد. در این مدل، کاربران می‌توانند استثناءپذیری مجوزهایی را که اعطا می‌کنند تعیین نمایند. مجوزهای قوی باید همیشه اعمال شده و نمی‌توانند همپوشانی داشته باشند، ولی مجوزهای ضعیف ممکن است به وسیلهٔ مجوزهای قوی و ضعیف دیگر همپوشان گردند (به بخش ۴-۱ مراجعه شود).

فرض کنید U مجموعهٔ کاربران، G مجموعهٔ گروهها، $S = U \cup G$ مجموعهٔ عاملها و P مجموعهٔ اَعمال قابل اجـرا روی جداول باشد. بر این اساس یک مجوز به صورت زیر تعریف می‌شود (که توسعه‌یافتهٔ تعریف مجوز در [۱۱] است):

تعریف ۱ (مجوز). مجوز a یک شش‌تایی $<s, p, pt, t, g, at>$ است که:

- $s \in S$ عاملی است که مجوز به آن اعطا شده است.
- $p \in P$ عملی است که مجوز برای اجرای آن داده شده است.
- $pt \in \{+, -\}$ بیانگر مجوز مثبت یا منفی است.
- $t \in T$ جدولی است که مجوز برای دسترسی به آن بیان گردیده است (اگر $pt = -$ باشد، $t \in BT$ است).
- $g \in U \cup \{system\}$ که معرف اعطاکنندهٔ مجوز است (نماد system در بخش ۵ توضیح داده شده است).
- $at \in \{strong, weak\}$ که استثناءپذیری مجوز را مشخص می‌کند.

مجموعهٔ مجوزهایی که همزمان روی جداول وجود دارد، *حـالت کنـترل دسترسـی* سیسـتم نـامیده می‌شـود. ایـن حـالت را می‌توان با استفاده از گراف نمایش داد [۲]. اگر عامل s_i یک مجوز مثبـت (منفـی) بـرای عمـل p روی جـدول t داشـته باشـد، سه‌تایی $<p, +, t>$ ($<p, -, t>$) به گره s_i در گراف عضویت نسبت داده می‌شود. برای بیان مجوزهـای قـوی نـیز از حـروف پررنگ استفاده می‌گردد. مثالی از گراف حالت کنترل دسترسی، در شکل ۱ آمده است. در این شکل، T_iهـا بیـانگر جـداول و V_7 دیدی است که بر پایهٔ جدول T_7 تعریف شده است. در ادامهٔ مقاله برای سادگی چنانچه اعطاکنندهٔ مجوز مهم نباشـد یـک مجوز، با چهارتایی $<s, p, pt, t>$ و به صورت پررنگ یا معمولی نمایش داده می‌شود.

مجوزهای مالک (ایجادکنندهٔ) یک دید، همان مجوزهای مثبت و معتبر (بخش ۴-۲) وی روی جداول ارجاع داده شـده در تعریف دید است (چرا که در غیر این صورت کاربر می‌تواند با تعریف دید جدید، محدودیتهای موجود در دسترسـی بـه یـک جدول را نقض کند). این مجوز، قوی می‌باشد اگر مالک دید روی هر جدولی که به طور مستقیم در تعریف دید به آن ارجـاع داده شده است، یک مجوز قوی (برای آن عمل) داشته باشد و در غیر این صورت، ضعیف خواهد بود. چون مجوزهای مـالک دید وابسته به مجوزهای وی روی جدولهای ارجـاعی در تعریـف دیـد اسـت، تغییـر در مجوزهـای روی جـداول، بـاعث تغییـر مجوزهای مربوط به دید می‌گردد که الگوریتمهای مربوط به این تغییرات در مراجع [۱۰ و ۱۲] آمده است.

٤- مدل کنترل دسترسی جدید

در این مدل می‌توان مجوزها را هم برای کاربر و هم برای گروه تعیین کرد. یک کاربر در کنار مجوزهای مسـتقیم خـود، تمـام مجوزهای گروههایی را که به آنها متعلق است در اختیار دارد. استفاده از مجوز قوی و ضعیف باعث افزایش قـدرت ایـن مـدل شده و در سیاست مدیریتی غیر متمرکز، مجوز قوی تنها ابزار اطمینان کاربران از اِعمال حتمی مجوزهای تعیین شدهٔ آنان اسـت. ویژگیهای این مدل را در موارد زیر می‌توان خلاصه کرد:

- مجوزها می‌توانند مثبت یا منفی و قوی یا ضعیف باشند.
- مجوزهای قوی همواره باید اعمال شوند، پس یک عامل نمی‌تواند همزمان دو مجوز قوی، یکی مثبت و دیـگری منفـی برای یک عمل روی جدولی داشته باشد.
- اگر عاملی دو مجوز مثبت و منفی، یکی قوی و دیگری ضعیف برای یک عمل روی جدولی داشته باشد مجـوز قـوی، مجوز ضعیف را می‌پوشاند.

- اگر عاملی دو مجوز ضعیف، یکی مثبت و دیگری منفی برای یک عمل روی جدولی داشته باشد، قانون همپوشانی توسط روابط عضویت بین این عامل و عاملهای دارندهٔ این مجوزها تعیین می‌شود.
- یک عامل به جدولی می‌تواند دسترسی داشته باشد که برای آن یا مجوز مثبت قوی و یا مجوز مثبت ضعیف بدون تصادم و همپوشانی داشته باشد.

٤-١- تعریف صوری مدل

بر اساس نوع مجوز، هرگاه دو مجوز قوی و ضعیف در تضاد (یکی مثبت و دیگری منفی و هر دو روی یک جدول) باشند تقدم با مجوز قوی است. همچنین واضح است که دو مجوز قوی و متضاد همزمان نمی‌تواند وجود داشته باشد، ولی زمانی که دو مجوز ضعیف در تضاد باشند چه باید کرد. در این حالت، مجوزهای مستقیم عامل، نسبت به مجوزهای گروههایی که به آن تعلق دارد تقدم بالاتری دارد. این تقدم فقط باید در مسیر عضویت بین عامل و گروههای متعلق به آن در نظر گرفته شود زیرا یک عامل از مسیرهای مختلفی می‌تواند عضو یک گروه باشد. همپوشانی مجوزها بر این اساس تعریف می‌گردد.

تعریف ۲ (همپوشانی مجوز). دو مجوز a_i و a_j را با شرایط $p(a_i) = p(a_j)$، $pt(a_i) \neq pt(a_j)$ و $t(a_i) =(\hookleftarrow)\ t(a_j)$ در نظر بگیرید. از نظر عامل s (که در شرایط $s \in_n s(a_i)$، $s \in_m s(a_j)$ و $n, m \geq 0$ صدق می‌کند) مجوز a_i، مجوز a_j را در مسیر $mp \in MP(s, s(a_j))$ می‌پوشاند ($a_i \rangle_s^{mp} a_j$) اگر و فقط اگر یکی از شرایط زیر برقرار باشد:

- $at(a_i) = \text{"strong"}$ و $at(a_j) = \text{"weak"}$.
- $at(a_i) = at(a_j)$، $s(a_i) \neq s(a_j)$ و $s(a_i)$ در مسیر mp موجود باشد.

اگر مجوز a_i، مجوز a_j را در همهٔ مسیرهای عضویت s به $s(a_j)$ پوشش دهد برای سادگی از نماد $a_i \rangle_s a_j$ استفاده می‌شود.

مثال ۱. با توجه به شکل ۱ روابط همپوشانی زیر برقرار است:

<**Non-citizens, sel, -, T_1**> $\rangle_{Nima}$ <Employees, sel, +, T_1>;
<**Users, sel, +, T_6**> $\rangle_{Sima}$ <Sima, sel, -, T_6>;
<**Omid, sel, +, T_2**> $\rangle_{Omid}$ <Consultants, sel, -, T_2>;
<**ConsC, sel, +, T_3**> $\rangle_{Ali}$ <Consultants, sel, -, T_3>;
<**Non-citizens, sel, -, T_7**> $\rangle_{Mona}$ <Mona, sel, +, V_7>;
<Ali, sel, -, T_7> $\rangle_{Ali}$ <ConsC, sel, +, V_7>;
<**Res2, sel, -, T_4**> $\rangle_{Amir}^{mp}$ <Soft-developers, sel, +, T_4>, *mp* = <Amir, Res2, Researchers, Soft-developers>.

تعریف ۳ (مجوز قابل اعمال). مجوز a_j به عامل $s \in_n s(a_j)$ $(n \geq 0)$ قابل اعمال است اگر و فقط اگر مسیر عضویتی از s به $s(a_j)$ موجود باشد به گونه‌ای که a_j در این مسیر از نظر عامل s همپوشانی نداشته باشد.

مثال ۲. شکل ۱ را در نظر بگیرید:

مجوز <Soft-developers, sel, +, T_3> به Amir قابل اعمال است،
مجوز <Soft-developers, sel, +, T_4> به Amir قابل اعمال نیست،
مجوز <Employees, sel, +, T_1> به Nima قابل اعمال نیست،
مجوز <ConsC, sel, +, V_7> به Ali قابل اعمال نیست.

تعریف ۴ (مجوزهای تصادم‌دار). دو مجوز a_i و a_j که هر دو به عامل s قابل اعمال هستند، تصادم دارند ($a_i \lozenge_s a_j$) اگر و فقط اگر $p(a_i) = p(a_j)$، $pt(a_i) \neq pt(a_j)$ و یکی از شرایط زیر برقرار باشد:

- $at(a_i) = at(a_j) = \text{"strong"}$ و $(t(a_i) \hookleftarrow t(a_j) \vee t(a_i) = t(a_j))$.
- $at(a_i) = at(a_j) = \text{"weak"}$ و $t(a_i) = t(a_j)$.

مثال ۳. با توجه به شکل ۱ برخی از مجوزهای تصادم‌دار عبارتند از:

<Soft-developers, sel, +, T_3> $\Diamond_{Amir}$ <Consultants, sel, -, T_3>;
<Researchers, sel, +, T_2> $\Diamond_{Amir}$ <Consultants, sel, -, T_2>;
<Non-citizens, sel, -, T_7> $\Diamond_{Nima}$ **<Staff, sel, +, V_7>**.

بر اساس این تعریف، یک عامل ممکن است به دیدی دسترسی داشته باشد هر چند که به جداول ارجاعی آن دسترسی ندارد. دلیل این امر آن است که یک دید دارای بخشی از داده‌های جداولی می‌باشد که بر پایهٔ آنها تعریف گردیده است. باید دقت کرد که اگر چه مجوزهای مثبت دید بر مجوزهای منفی جداول حقیقی برتری دارد ولی این به معنای همپوشانی نیست. مثلا در شکل ۱، کاربران Roya ، Vahid ، Omid و Reza اجازهٔ دسترسی به دید V_7 را با عمل select دارند ولی نمی‌توانند به جدول T_7 با همین عمل دسترسی داشته باشند. استدلال فوق فقط در حالتی که مجوز منفی جدول پایه، از نوع ضعیف باشد برقرار است.

تعریف ۵ (سازگاری حالت کنترل دسترسی). حالت کنترل دسترسی، *سازگار* است اگر و فقط اگر از نظر هیچ عاملی، مجوزهای قوی باهم تصادم نداشته باشند.

اطمینان از سازگاری مجوزها، وظیفهٔ سیستم کنترل دسترسی است و با انجام عملیاتی از قبیل تغییر مجوز (مانند اعطا یا لغو مجوز) یا تغییر گراف عضویت گروه (مانند افزودن یا حذف اعضای گروهها) در صورتی موافقت می‌شود که حالت کنترل دسترسی حاصل، سازگار باشد. در این مدل، تصادم مابین مجوزهای ضعیف پذیرفته می‌شود زیرا این مجوزها استثناءپذیر بوده و تصادم همواره قابل رفع است. کاربران می‌توانند یا آشکارا و به اختیار خود و یا توسط سیاست از پیش تعیین شدهٔ سیستم در زمان دسترسی، تصادم را رفع کنند. این سیاست پیش‌فرض، تقدم را بر پایهٔ نفی قرار می‌دهد.

٤-۲- الگوریتم کنترل دسترسی

در این مدل، کنترل دسترسی بر پایهٔ مفهوم «مجوز معتبر» استوار می‌باشد که در زیر تعریف شده است.

تعریف ۶ (مجوز معتبر). مجوز a برای عامل s (با شرط $s \in_n s(a)$) معتبر است اگر به s قابلِ اعمال بوده و با مجوزهای دیگر آن تصادم نداشته باشد.

یک درخواستِ دسترسی پذیرفته می‌شود اگر و فقط اگر مجوز مثبت و معتبر برای آن وجود داشته باشد. در این مقاله، کنترل دسترسی به صورت تابع $access()$ با پارامتر ورودی $<u, p, t>$ در نظر گرفته می‌شود که بیانگر درخواست عامل u برای اجرای عمل p روی جدول t است. اگر این درخواست مجاز باشد، مقدار $true$ و در غیر این صورت مقدار $false$ توسط این تابع بازگردانده می‌شود. تابع $access()$ به صورت زیر تعریف می‌شود:

$$access(u,p,t) = \begin{cases} strong_auth(u,p,t) & if\ strong_auth \neq undecided \\ weak_auth(u,p,t) & otherwise \end{cases}$$

که تعریف توابع $strong_auth$ و $weak_auth$ بدین صورت است:

$$strong_auth(u,p,t) = \begin{cases} true & if\ exists\ a\ strong\ positive\ privilege\ for\ the\ access \\ false & if\ exists\ a\ strong\ negative\ privilege\ for\ the\ access \\ undecided & otherwise \end{cases}$$

$$weak_auth(u,p,t) = \begin{cases} true & if\ exists\ a\ weak\ positive\ privilege\ for\ the\ access\ valid\ for\ u \\ false & otherwise \end{cases}$$

روش کار تابع ()*strong_auth* . این تابع، ابتدا مجوزهای قوی کاربر درخواست‌کننده را جستجو می‌کند. اگر چنین مجوزی وجود داشت (مثبت یا منفی)، پاسخ متناسب (*true* یا *false*) برگردانده می‌شود. در غیر این صورت مجوزهای گروههایی که کاربر عضو آنهاست جستجو می‌گردد و در صورت یافتن مجوزی، مانند حالت قبل جواب مناسب تولید می‌شود. اگر هیچ مجوزی یافت نگردید، تابع مقدار *undecided* را باز می‌گرداند. اگر درخواست کاربر برای دسترسی به یک دید باشد علاوه بر مجوزهای دید، مجوزهای منفی جداول حقیقی ارجاع داده شده در تعریف آن دید نیز کنترل می‌شود. نیازی به بررسی تمام مجوزها نیست و اجرای تابع زمانی خاتمه می‌یابد که یک مجوز قوی (مثبت یا منفی) پیدا شود.

روش کار تابع ()*weak_auth* . ابتدا حالتی را در نظر بگیرید که درخواست برای یک جدول حقیقی است. مجوزهای منفیِ کاربرِ درخواست‌کننده جستجو می‌گردد و در صورت یافتن مجوز منفی، مقدار *false* برگردانده شده و کار خاتمه می‌یابد. در غیر این صورت، مجوزهای مثبت وی کنترل شده و اگر مجوز مثبتی یافت شد با بازگرداندن مقدار *true* اجرای تابع به پایان می‌رسد (دلیل این امر که ابتدا مجوزهای منفی و سپس مجوزهای مثبت جستجو می‌گردد، امکان وجود تصادم مابین مجوزهای ضعیف است). اگر هیچ مجوزی برای کاربر یافت نشد، مجوزهای گروههایی که کاربر عضو آنهاست بررسی می‌گردد. این گروهها با کمک الگوریتم BFS[۵] روی گراف عضویت گروه (با شروع از گره کاربر) پیمایش می‌شوند. اگر یک مجوز مثبت در گروهی پیدا شد پیمایش در آن شاخه متوقف می‌گردد چراکه این مجوز، مجوزهای گروههای سطح بالاتر خود را می‌پوشاند. اجرای تابع یا با یافتن مجوز منفی و یا با پایان پیمایش گراف خاتمه می‌یابد. در حالت اول مقدار *false* برگردانده می‌شود و در حالت دوم نیز اگر هیچ مجوز مثبتی پیدا نشده باشد، مقدار *false* و در غیر این صورت مقدار *true* خروجی آن خواهد بود.

اکنون حالتی را در نظر بگیرید که درخواست برای دسترسی به یک دید است. ابتدا مجوزهای مثبتِ کاربرِ درخواست‌کننده روی دید جستجو می‌گردد و در صورت یافتن چنین مجوزی، مقدار *true* برگردانده شده و کار خاتمه می‌یابد. در غیر این صورت، مجوزهای منفی وی روی جداول حقیقی ارجاع داده شده در تعریف دید، کنترل شده و اگر مجوز منفی یافت شد با بازگرداندن مقدار *false* اجرای تابع به پایان می‌رسد (دلیل این امر که بر خلاف حالت قبل، ابتدا مجوزهای مثبت جستجو می‌گردد معتبر بودن مجوزهای مثبت برای دید حتی در حضور مجوزهای منفی برای جداول حقیقی است). اگر هیچ مجوزی برای کاربر یافت نشد، مجوزهای گروههایی که کاربر عضو آنهاست بررسی می‌گردد. این گروهها با کمک الگوریتم BFS روی گراف عضویت گروه (با شروع از گره کاربر) پیمایش می‌شوند. برای هر گروه، وجود مجوزهای مثبت برای دید و مجوزهای منفی برای جداول حقیقی ارجاع داده شده در تعریف دید بررسی می‌شود. اگر مجوز منفی در گروهی پیدا شد پیمایش در آن شاخه متوقف می‌گردد چراکه این مجوز، مجوزهای گروههای سطح بالاتر خود را می‌پوشاند. اجرای تابع یا با پایان پیمایش گراف و یا با یافتن مجوز مثبت خاتمه می‌یابد. در حالت اول *false* و در حالت دوم *true* برگردانده می‌شود.

۵- مدیریت مجوزها

کاربر ایجاد کنندهٔ یک شئ، مالک آن محسوب شده و تنها شخصی است که می‌تواند آن را حذف کند. زمانی که کاربری یک جدول حقیقی را ایجاد می‌کند، برای تمامی اَعمال قابل اجرا روی آن جدول از طرف سیستم (system)، مجوزهای مثبت و قوی اعطا می‌گردد که این مجوزها با حذف جدول لغو می‌شوند. یک کاربر به عنوان مالک می‌تواند علاوه بر مجوزهای (قوی یا ضعیف و مثبت یا منفی) خود، مجوزهای مدیریتی را نیز به کاربران دیگر اعطا کند.

---

[۵] Breath First Search

تعریف ۷ (مجوز مدیریتی). شش‌تایی $<s, p, ap, gat, t, g>$ ، یک مجوز مدیریتی است که در آن:

- $s \in S$ عاملی است که مجوز به آن اعطا شده است.
- $p \in P$ عملی است که مجوز برای مدیریت آن بیان شده است.
- $ap \in$ {adm-access, administer} بیانگر نوع مجوز است.
- $gat \in$ {strong, weak} نوع مجوزی است که s می‌تواند اعطا کند.
- $t \in T$ جدولی است که مجوز برای مدیریت دسترسی به آن بیان شده است.
- $g \in U$ که معرف اعطاکنندهٔ مجوز است.

در این مدل، دو نوع مجوز مدیریتی وجود دارد: adm-access و administer . در صورتی که کاربری مجوز مدیریتی adm-access را برای عمل p روی جدول t داشته باشد، می‌تواند مجوز مثبت یا منفی عمل p روی جدول t را به عاملهای دیگر اعطا (لغو) کند. ولی اگر کاربری، مجوز مدیریتی administer را برای عمل p روی جدول t داشته باشد، می‌تواند علاوه بر اعطای (لغو) مجوز مثبت یا منفی عمل p، مجوزهای مدیریتی مربوط به آن را نیز به عاملهای دیگر اعطا (لغو) کند.

همراه با هر مجوز مدیریتی نوع قوی یا ضعیف نیز ذکر می‌گردد که البته این مربوط به خود مجوز مدیریتی نیست، بلکه نوع مجوزهایی است که به وسیلهٔ آن می‌تواند اعطا یا لغو گردد. در واقع، مجوزهای مدیریتی به دو دستهٔ قوی و ضعیف تقسیم نشده و نمی‌توانند همپوشانی داشته باشند. می‌توان این مجوزها را همواره قوی فرض کرد، چرا که در صورت همپوشانی مجوزهای مدیریتی ممکن است مجوزهای اعطا شده‌ای در سیستم موجود باشد که مجوزهای مدیریتی آنها از بین رفته است. یعنی در حالی که مجوزهای مدیریتی کاربری معتبر نیست، مجوزهای اعطایی وی هنوز معتبر باشد. انتشار اثر همپوشانی، مدیریت مجوزها را بسیار پیچیده می‌سازد، زیرا در رابطهٔ همپوشانی نه تنها ساختار گروهها، بلکه اعطاکنندگان مجوزها نیز باید در نظر گرفته شود.

واضح است که اگر عاملی مجوز انجام عملی را روی جدولی نداشته باشد، وجود مجوز مدیریتی آن عمل برای خود عامل منطقی نیست. بنابراین لازم است که عاملهای دارندهٔ مجوز مدیریتی یک عمل، مجوز قوی و منفی را برای آن عمل نداشته باشند. در این مدل مجوزهای مدیریتی مثبت ومنفی از یکدیگر تفکیک نشده‌اند و اگر کاربری یک مجوز مدیریتی داشته باشد می‌تواند از آن برای اعطا یا لغو مجوز به هر دو صورت مثبت یا منفی استفاده کند.

زمانی که یک دید تعریف می‌شود، مجوزهای مدیریتی آن از مجوزهای جداول ارجاع داده شده در تعریف دید تعیین می‌شود. به منظور داشتن مجوز مدیریتی برای عملی روی دید، کاربر باید یک مجوز مدیریتی برای آن عمل در هر جدول ارجاعی دید داشته باشد. کاربر در صورتی مجوز مدیریتی administer را برای عملی روی یک دید خواهد داشت که روی تمامی جداول ارجاعی آن دید، مجوز administer را برای آن عمل داشته باشد. اگر حداقل یکی از این مجوزها، adm-access باشد مجوز مدیریتی دید نیز از نوع adm-access خواهد بود.

مجوزها فقط توسط کاربرانی که آنها را اعطا کرده‌اند لغو می‌شود. همچنین کاربری می‌تواند اعطاکنندهٔ مجوز عملی روی یک جدول باشد که مجوز مدیریتی آن عمل را داشته باشد. در نتیجه، زمانی که مجوز مدیریتی کاربر برای عملی لغو می‌شود، لغو بازگشتی مجوزهای اعطا شده توسط وی یا به ارث رسیده برای کاربران دیدها لازم است. الگوریتمهای لغو که به صورت بازگشتی عمل می‌کنند در مدلهای دیگر نیز وجود داشته و می‌توان از آنها در اینجا استفاده کرد [۱۰، ۱۱، ۱۳ و ۱۴]. البته روشهای غیربازگشتی نیز وجود دارد. مثلاً اگر به هنگام لغو، نیاز به حذف بیش از یک مجوز باشد عمل لغو اجازه داده نمی‌شود [۱۵]، یا این که مجوزهای اعطاشده توسط کاربری که مجوز مدیریتی آن لغو شده، حذف نگردد بلکه اعطاکنندهٔ جدیدی برای آنها تعیین شود [۱۱].

٦- سازگاری و رفع تصادم

با اجرای عملیات مدیریتی توسط کاربران، مجموعهٔ مجوزهای قابلِ اعمال به عاملها ممکن است تغییر کند. این عملیات می‌تواند باعث تغییر حالت کنترل دسترسی (مانند اعطا و لغو مجوز عملی روی جدول)، تغییر گراف عضویت گروه (مانند درج و حذف اعضای گروه) و تغییر جدولها (مانند ایجاد و حذف جدولها) شود. اگر چه برخی از این عملیات هیچ اثری در خود مجوزها ندارند ولی می‌توانند در مجوزهای قابلِ اعمال اثر گذاشته و منجر به بروز تصادم شوند. مثلاً اگر کاربری به گروهی اضافه شود، مجوزهای گروه به کاربر قابلِ اعمال می‌شود.

همانطور که در بخش ۴ بیان شد، در این مدل مجوزهای قوی نباید تصادم داشته باشند ولی وجود مجوزهای ضعیف تصادم‌دار ایرادی ندارد. در این بخش شرح داده می‌شود که چگونه عملیات مدیریتی می‌تواند سازگاری حالت کنترل دسترسی را تحت تأثیر قرار داده و کاربران به چه شکل می‌توانند تصادمهای مابین مجوزهای ضعیف را بر حسب نیازشان رفع کنند.

٦-١- اطمینان از سازگاری حالت کنترل دسترسی

واضح است اَعمالی از قبیل لغو مجوزها، حذف اعضای گروه و حذف جدولها که باعث کاهش مجوزهای قابلِ اعمال به عاملها می‌گردند، در سازگاری سیستم اثری ندارند. ولی اَعمالی مانند ایجاد جداول جدید، اعطای مجوزهای قوی و درج اعضای جدید در گروهها می‌توانند منجر به ناسازگاری حالت کنترل دسترسی شوند که در ادامه، هر کدام از آنها بررسی می‌گردند.

ایجاد جداول جدید. در بخشهای ۴ و ۵ توضیح داده شد که چگونه به هنگام ایجاد یک جدول جدید، مجوزها به مالک آن تخصیص داده می‌شود. این مجوزهای جدید هیچ اثری در سازگاری سیستم ندارند، یعنی اگر به یک حالت سازگار اِعمال شوند همواره یک حالت سازگار جدید به وجود می‌آورند. اثبات این امر نیز ساده است: یک کاربر در صورتی مجوز قوی روی دید خواهد داشت که برای همهٔ جداول ارجاعی در تعریف دید، یک مجوز قوی داشته باشد. بنابراین اگر مجوز جدید، تصادم دار باشد باید مابین مجوزهای موجود روی جداول ارجاعی در تعریف دید نیز تصادمی وجود داشته باشد که این متناقض با فرض اولیهٔ سازگاری حالت کنترل دسترسی قبل از اجرای عمل است.

اعطای مجوزهای قوی. این عمل می‌تواند باعث بروز ناسازگاری گردد، مثلاً چنانچه مجوز <**sel, +, T_4**> به گروه ConsA در شکل ۱ اعطا شود ناسازگاریهای زیر به وجود می‌آید:

<**ConsA, sel, +, T_4**> $\Diamond_{Amir}$ <**Res2, sel, -, T_4**>;
<**ConsA, sel, +, T_4**> $\Diamond_{ConsA}$ <**Consultants, sel, -, T_4**>.

الگوریتم کنترل سازگاری به هنگام درج مجوز قوی و جدید a بدین صورت عمل می‌کند: ابتدا مجوزهای عامل $s(a)$ (یا عضوی از آن) و یا هر گروهی که شامل $s(a)$ (یا عضوی از آن) است بررسی می‌گردد. سپس، مجموعهٔ مجوزهای قوی برای عمل $p(a)$، روی جدول t با یکی از شرایط $t = t(a)$ ، $t(a) \lrcorner\!\leftarrow t$ یا $t \lrcorner\!\leftarrow t(a)$ و متفاوت با $pt(a)$ تعیین می‌شود. چنانچه این مجموعه، تهی باشد هیچ تصادمی وجود نداشته و خروجی الگوریتم مقدار ok خواهد بود. در غیر این صورت برای هر مجوز یافت شده، تصادمهای موجود برگردانده می‌شود. اگر دو مجوز، هم روی یک گروه و هم روی یکی از اعضای آن با هم تصادم داشته باشند فقط تصادم روی گروه بازگردانده می‌شود.

درج اعضای جدید در گروهها. این عمل می‌تواند باعث بروز ناسازگاری گردد، برای مثال در شکل ۱ چنانچه کاربر Mina به طور موقت عضو گروه Non-citizens شود، ناسازگاری زیر به وجود می‌آید:

<**ConsB, sel, +, V_7**> $\Diamond_{Mina}$ <**Non-citizens, sel, -, T_7**>.

الگوریتم کنترل سازگاری سیستم به هنگام درج عضو m در گروه G_k بدین صورت است: تمام مجوزهای G_k یا گروههای شامل آن (فاقد عضو m) با مجوزهای (۱) عاملهای شامل m که عضو G_k نیستند (همراه با خود m) و (۲) گروههای شامل عناصری از m که نه عضو m، نه عضو G_k و نه شامل G_k (یا یکی از اعضای آن) هستند مقایسه می‌شود. اگر دو مجوز با یک عمل، یکی مثبت و دیگری منفی و روی یک جدول (یا جدول یکی، ارجاع داده شده در جدول دیگری باشد) یافت شد تصادمِ تشخیص داده شده، خروجی الگوریتم خواهد بود. اگر دو مجوز، هم روی یک گروه و هم روی یکی از اعضای آن باهم تصادم داشته باشند فقط تصادم روی گروه بازگردانده می‌شود.

٦-۲- رفع تصادم مابین مجوزهای ضعیف

در صورتی که همزمان چند مجوز ضعیفِ تصادم‌دار برای یک دسترسی موجود باشد این مدل، دسترسی را نفی می‌کند. ولی استفاده از این سیاست پیش‌فرض، تنها راه حل برای رفع تصادم بین مجوزهای ضعیف نمی‌باشد و کاربران می‌توانند تصادم را با درج مجوزهای جدید در حالت کنترل دسترسی رفع کنند. به عبارت دیگر کاربران مجبور به پذیرش سیاست پیش‌فرض سیستم نبوده و می‌توانند تصادم را مطابق نیاز خود از بین ببرند.

هر تصادم (جز در یک حالت خاص) می‌تواند با افزودن یک مجوز جدید رفع شود. این حالت خاص زمانی رخ می‌دهد که عاملهای مجوزهای تصادم‌دار یکسان باشند. فرض کنید دو مجوز a_i و a_j از نظر عامل s تصادم داشته باشند، بنابراین طبق تعریف ۴، $t(a_i) = t(a_j)$ ، $p(a_i) = p(a_j)$ و $pt(a_i) \neq pt(a_j)$ است. سه حالت زیر می‌تواند رخ دهد:

حالت ۱: $s(a_i) \neq s(a_j)$. واضح است که $s \neq s(a_i)$ و $s \neq s(a_j)$ (چرا که در غیر این صورت a_i و a_j از نظر عامل s تصادم ندارند). این تصادم می‌تواند با افزودن مجوز a_m با شرایط $p(a_m) = p(a_i)$ ، $t(a_m) = t(a_i)$ و $s(a_m) = s$ رفع شود. مقدار $pt(a_m)$ می‌تواند + یا - باشد بسته به اینکه اعطاکنندهٔ مجوز با چه هدفی آن را اعطا می‌کند.

حالت ۲: $s(a_i) = s(a_j)$ *ولی* $s \neq s(a_i)$. تصادم می‌تواند همانند حالت ۱ با افزودن مجوز جدید a_m رفع شود.

حالت ۳: $s(a_i) = s(a_j) = s$. در این حالت دو روش برای رفع تصادم وجود دارد: یا حذف یکی از دو مجوز و یا افزودن یک مجوز قوی و جدید a با شرایط $s(a) = s$ ، $p(a) = p(a_i)$ و $t(a) = t(a_i)$. البته درج مجوز قوی همواره امکان‌پذیر نیست چرا که می‌تواند خود باعث ناسازگاری حالت کنترل دسترسی شود. بنابراین حذف یکی از دو مجوز تصادم‌دار بهترین راه حل می‌باشد (وجود همزمان دو مجوز مثبت و منفی تصادم‌دار برای یک عامل، اساساً بی‌معنی است).

مثال۴. در شکل ۱ تصادم زیر مابین دو مجوز ضعیف وجود دارد:

<Researchers, sel, +, T_2> $\Diamond_{Amir}$ <Consultants, sel, -, T_2>.

این تصادم در شرایط حالت ۱ صدق کرده و برای رفع آن کافی است که مجوز <Amir, sel, (+ or -), T_3> به حالت کنترل دسترسی سیستم اضافه شود. مثبت یا منفی بودن مجوز به هدف اعطاکنندهٔ آن بستگی دارد.

۷- نتیجه‌گیری

در این مقاله یک مدل کنترل دسترسی برای پایگاههای دادهٔ رابطه‌ای ارائه شده است که می‌تواند برای بیان سیاستهای مختلف (سیاستهای باز یا بسته و سیاستهای تقدم نفی یا تقدم استثناء) به کار رود. در این مدل، امکان استفادهٔ توأم از مجوزهای مثبت یا منفی و قوی یا ضعیف (استثناءپذیری مجوز) وجود دارد. همچنین از مدیریت مجوزها در هر سطحی از سلسله‌مراتب گروه همراه با تعلیق موقتی آنها پشتیبانی می‌شود. مدیریت این مدل، ترکیبی از دو سیاست مالکیت و غیرمتمرکز است و مجوزهای

مدیریتی می‌تواند به گونه‌ای واگذار گردد که کاربران بتوانند کنترل مجوزها را روی جداول تحت مالکیت خود حفظ کنند. همچنین الگوریتمهای کنترل سازگاری سیستم همراه با نحوهٔ رفع تصادم مابین مجوزهای ضعیف شرح داده شده است.

اگر چه این مدل بر پایهٔ سیستمهای مدیریت پایگاه دادهٔ رابطه‌ای بنا شده است ولی می‌توان با کمی تغییر در سایر سیستمهای مدیریت داده نیز به کار برد. کار بعدی افزودن محدودیتهای زمانی به این مدل است. همچنین ویژگیهای این مدل می‌تواند برای پایگاههای دادهٔ مؤتلفه‌ای[۶] که به بیان سیاستهای امنیتی مختلف با یک مدل کنترل دسترسی مشترک نیاز دارند، استفاده شود.

مراجع

[1] R. Sandhu, and P. Samarati, "Access Control: Principles and Practice", *IEEE Communications Magazine,* September, 1994, pp. 40-48.

[2] E. Bertino, S. Jajodia, and P. Samarati, "A Flexible Authorization Mechanism for Relational Data Management Systems", *ACM Transactions on Information Systems,* Vol. 17, No. 2, 1999, pp. 101-140.

[3] S. Jajodia, P. Samarati, and V. S. Subrahmanian, "A Logical Language for Expressing Authorizations", In *Proceedings of IEEE Symposium on Security and Privacy,* (Oakland, CA, MAY), IEEE Press, Piscataway, NJ, 1997, pp. 31-42.

[4] M. Satyanarayanan, "Integrating Security in Large Distributed System", *ACM Transactions on Computer Systems,* Vol. 7, No. 3, 1989, pp. 247-280.

[5] T. F. Lunt, "Access Control Policies for Database Systems", In *Database Security II: Status and Prospects,* C. E. Landwehr, Ed. North-Holland Publishing Co., Amsterdam, T he Netherlands, 1989, pp. 41-52.

[6] F. Rabitti, E. Bertino, W. Kim, and D. Woelk, "A Model of Authorization for Next-Generation Database Systems", *ACM Transactions on Database Systems,* Vol. 16, 1991, pp. 88-131.

[7] H. Bruggemann, "Rights in an Object-Oriented Environment", In *Database Security V: Status and Prospects,* C. Landwehr and S. Jajodia, Eds. Elsevier North-Holland, Inc., NewYork, NY, 1992, pp. 99-115.

[8] H. Shen and P. Dewan, "Access Control for Collaborative Environments", In *Proceedings of the International Conference on Computer Supported Cooperative Work,* ACM Press, New York, NY, 1992, pp. 51-58.

[9] N. Gal-Oz, E. Gudes, and E. B. Fernandez, "A Model of Methods Authorization in Object-Oriented Databases", In *Proceedings of the International Conference on Very Large Data Bases,* (Dublin, Irland), Morgan Kaufmann Publishers Inc., San Francisco, 1993, pp. 52-61.

[10] P. G. Griffiths, and B. Wade, "An Authorization Mechanism for a Relational Database System", *ACM Transactions on Database Systems,* Vol. 1, No. 3, 1976, pp. 243-255.

[11] E. Bertino, P. Samarati, and S. Jajodia, "An Extended Authorization Model for Relational Databases", *IEEE Transactions on Knowledge and Data Engineering,* Vol. 9, No. 1 (Jan.-Feb.), 1997, pp. 85-101.

[12] P. G. Selinger, "Authorizations and Views", In *Distributed Data Bases,* I. W. Draffan and F. Pooe, Eds. Cambridge University Press, New York, NY, 1990, pp. 233-246.

[13] R. Fagin, "On an Authorization Mechanism", *ACM Transactions on Database Systems,* Vol. 3, No. 3, 1978, pp. 310-319.

[14] Informix, *Informix-Online/Secure. Security Features User's Guide,* Informix Software, Inc., 1993.

[15] J. Melton, "ISO/ANSI Working Draft-Database Language SQL 2", *Technical Report ANSI X3H2-90-309,* ANSI, New York, NY, 1990.

---

[۶] federated databases