Revogação com Downgrade

Fábio Negrello, Jacques Wainer

Instituto de Informática – Universidade Estadual de Campinas (UNICAMP)
Campinas, 13083-970, SP, Brasil

negrello@cpqd.com.br, wainer@ic.unicamp.br

Abstract. *This paper presents a flexible authorization model that allows the precise control of delegation chains. This can be obtained both by defining the maximum lenght of delegation chains, and by associating constraints for the acceptance of delegations. In this model, we propose an efficient revocation with downgrade algorithm, that keeps the subjects with the biggest set of rights, considering the remaining alternative chains, after the revocation of delegations. Also, it is discussed an algorithm for determining the acceptance of new delegations, which verifies if there is at least one valid support chain for each new delegation.*

Resumo. *Este artigo apresenta um modelo flexível de autorização que permite controlar de forma precisa a formação de cadeias de delegações, tanto através da limitação no comprimento tais cadeias, como atráves da definição de condições para aceitação de novas delegações. Direcionado a este modelo, é proposto um algoritmo eficiente de revogação com downgrade, que mantém os sujeitos com o maior conjunto possível de direitos, considerando-se as cadeias de suporte alternativas restantes após a revogação de delegações. É apresentado um algoritmo para aceitação, que verifica se há ao menos uma cadeia de suporte válida para cada nova delegação.*

1. Introdução e Motivação

Entre as principais políticas de controle de acesso atualmente utilizadas destacam-se a política de controle de acesso discricionária (DAC) [Samarati and di Vimercati 2001] e a política baseada em papéis (RBAC). Entre os tipos de sistemas que normalmente utilizam a política DAC, embora não de forma isolada, estão os sistemas de gerenciamento de bases de dados (DBMS).

O modelo de autorização proposto por Griffts e Wade [Griffiths and Wade 1976], relacionado ao framework de bancos de dados relacionais conhecido como System R [Astrahan 1976], foi utilizado como base para a maioria dos sistemas DBMS comerciais atuais. Este modelo, como proposto inicialmente, possuía caráter essencialmente discricionário e descentralizado. O criador, ou dono, de uma tabela podia conceder autorizações sobre esta tabela a outros dos sujeitos do sistema, através do comando *grant*. Opcionalmente, era permitido ao criador da tabela conceder o direito de administração sobre este objeto a outros sujeitos do sistema. Desta forma, os sujeitos com direito de administração, como o criador, podiam conceder autorizações, ou ainda direito de administração, a outros sujeitos do sistema, formando assim um cadeia de delegações. Neste modelo foi também proposta uma forma de se revogar autorizações anteriormente concedidas a sujeitos do sistema, através do comando *revoke*. Este comando, além de revogar a autorização

ao concedida ao sujeito especificado, também revogava as autorizações que passavam a não ter mais uma cadeia de suporte em virtude desta revogação. Ou seja, as autorizações que não são mais alcançáveis a partir do dono do objeto são também revogadas, além da própria autorização revogada.

Com base neste histórico, de uma forma geral os sistemas DBMS modernos utilizam o modelo discricionário de controle de acesso (DAC), seja de forma pura, ou em conjunto com os modelos RBAC. Assim, permitem utilizar o conceito de delegação, que é a concessão de autorizações de um sujeito a outro do sistema. Embora seja necessário que outros sujeitos, além do criador do objeto, tenham o privilégio de conceder autorizações, isto pode levar a dois problemas principais.

Primeiramente, as cadeias de autorizações podem crescer inadvertidamente, tornando difícil para o dono de um objeto controlar quais sujeitos possuem acesso a este objeto, e em que condições este acesso é autorizado. Isto é especialmente significativo em sistemas onde sujeitos, objetos e autorizações representam, respectivamente, pessoas, recursos, e processos reais dentro de uma empresa ou organização. Como exemplo, observe que um gerente deveria ser capaz de garantir que uma tarefa especialmente importante não seja delegada a funcionários em um nível hierárquico distante, digamos, dois níveis dele próprio, o que poderia tornar difícil o acompanhamento sobre a execução da tarefa.

Além disso, sujeitos com interesses conflitantes em relação ao dono do objeto podem receber autorizações sobre este objeto via uma cadeia de delegações. Uma solução comum para este problema, empregada pela maioria dos modelos de controle de acesso, é a utilização de autorizações negativas. Entretanto, empregando-se esta solução em um sistema discricionário, o dono de um objeto deve atribuir autorizações negativas a todos os sujeitos que não deseja que o acessem. Isto nem sempre é factível, especialmente se o número de sujeitos é grande, ou se novos usuários são acrescentados com uma freqüência alta. Outra abordagem utilizada com o objetivo de controlar quais sujeitos tem acesso a determinadas autorizações é a definição de regras, ou restrições, genéricas. Esta abordagem é utilizada principalmente em modelos avançados de controle de acesso que permitem a especificação de autorizações através de linguagens lógicas [Woo and Lam 1993].

Resumindo, além de ser possível transmitir direitos de administração a sujeitos em um sistema, são necessárias também formas de se controlar a profundidade das cadeias de delegações formadas a partir do dono de um objeto e, além disso, prover mecanismos de controle que permitam especificar as condições que os sujeitos devem satisfazer para que possam adquirir determinadas autorizações, evitando assim conflitos em relação ao dono do objeto.

Neste artigo, primeiramente, serão resumidamente apresentados os trabalhos principais na área de modelos de autorização e revogação. Em seguida será apresentada a base do modelo conceitual proposto. Serão tratadas questões relacionadas à revogação de delegações não condicionais, e será apresentado um algoritmo eficiente para revogação de tais delegações. Em seguida, este algoritmo será extendido para considerar delegações condicionais. Finalmente, serão discutidos os principais pontos a serem incorporados em versões futuras, bem como sua possível influência no modelo atualmente proposto.

2. Trabalhos Relacionados

Dentre os principais modelos de autorização os mais conhecidos são os modelos *Take-Grant* [A.K. Jones and Snyder. 1976], e *Action-Entity* [U. Bussolati and Martella. 1993], sendo o segundo basicamente um aprimoramento do primeiro com um conjunto de privilégios de administração, e suporte a predicados em autorizações. O modelo *Take-Grant* utiliza uma estrutura em forma de grafo para representar sujeitos e objetos, bem como autorizações. Um determinado arco rotulado com um ou mais direitos indica que o sujeito, representado pelo vértice fonte, é autorizado a exercer os direitos sobre o objeto, representado pelo vértice destino.

Na área de sistemas de gerenciamento de bases de dados, o modelo proposto por Griffts e Wade dentro do framework do System R, como citado neste artigo, foi utilizado como base na maioria dos sistemas comerciais utilizados atualmente. Dirigido à área de *e-Consent*, destaca-se o modelo baseado em grafos [Ruan and Varadharajan 2004], que suporta delegação de autorizações e, ao contrário do modelo *Take-Grant* possui a vantagem de suportar autorizações negativas, e possuir mecanismos para resolução de conflitos quando autorizações divergentes são atribuídas ao mesmo sujeito.

Na literatura é ampla a gama de trabalhos relacionados ao emprego de delegações em sistemas que utilizam a política RBAC. Em especial, o modelo RBDM0 [Barka and Sandhu. 2000] é proposto como uma extensão ao modelo RBAC para permitir a inclusão de delegações. Neste modelo, todas as autorizações associadas a um papel são delegadas, como um conjunto. O modelo de delegação proposto em [Zhang et al. 2003a] permite delegações na forma DLGT(User1, Role1, User2, Role2), onde User1, associado ao papel Role1, delega este papel ao usuário User2, o qual já se encontra associado ao papel Role2. Assim como o modelo RBDM0, as autorizações são delegadas em conjunto. A necessidade de delegações parciais, isto é, de somente um subconjunto das autorizações associadas a um papel, é considerada pelo modelo PBDM [Zhang et al. 2003b]. Este modelo traz a vantagem de permitir delegações parciais, além de permitir delegações de um papel a outro, bem como de usuário a outro. O modelo proposto em [Yao et al. 2001] utiliza o conceito de nomeação, ou designação. Um usuário nomeia outro usuário, através da passagem de uma credencial, permitindo que este usuário ative um ou mais papéis. A ativação de papéis é baseada em regras. Assim, é possível que usuário nomeie outro usuário, mesmo que o nomeador não possua as autorizações que o nomeado venha a receber. Este conceito é similar ao conceito de delegações fortes, que será apresentado neste artigo, na medida em que usuário delega a outro usuário autorizações que ele mesmo não possui.

Em relação a mecanismos de revogação, o modelo Griffts-Wade utiliza timestamps para identificar as dependências entre as delegações existentes no sistema. Quando uma delegação é revogada, todas as delegações dependentes, ou seja, conferidas depois desta sendo removida, e que não possuam outra delegação de suporte, são também removidas. O mecanismo de revogação proposto em [Bertino et al. 1997], denominado noncascading-revocation, é uma extensão ao modelo Griffts-Wade. Este mecanismo também utiliza o timestamp das autorizações como um modo de se determinar dependências entre as delegações existentes no sistema. Porém, esta extensão permite que as autorizações revogadas sejam automaticamente reatribuídas pelo sujeito que as revogou, evitando assim a revogação em cascata.

3. Modelo DW-RBAC

A seguir será apresentada um introdução dos conceitos principais do modelo proposto em [Wainer et al. 2005, Wainer et al. 2003], necessários para o entendimento do problema de revogação de delegações.

Uma autorização representa o direito, ou privilégio, de execução de determinada operação sobre um objeto ou recurso do sistema. Desta forma, seja *O* o conjunto de objetos, ou recursos, a serem autorizados no sistema, e F o conjunto de operações que podem ser executadas sobre estes objetos. Uma Autorização é então definida por uma dupla (o, f) tal que *o* pertence a *O* e *f* pertence a F.

Uma delegação, por sua vez, representa a ação através da qual um sujeito transfere a outro sujeito determinada autorização. Seja S o conjunto de sujeitos definidos no sistema, e A, o conjunto de autorizações possíveis. Então uma delegação é definida pela tripla (g, r, a), onde *g* e *r* pertencem a S, e *a* pertence a A. Desta forma, uma delegação significa a transferência de uma autorização *a* de um sujeito *g* a outro sujeito *r*. A delegação não implica na perda da autorização por parte do primeiro sujeito, mas sim que o receptor da delegação passar a possuir a autorização enquanto *g* a possuir.

A própria ação de delegar uma autorização é também considerada uma autorização. A autorização incondicional de delegar determinada autorização *a* é representada por ud(a), onde a por sua vez, como definido acima, é representada por (o, f). É possível a um sujeito delegar somente determinada autorização (a), ou delegar esta autorização mais a autorização de delegá-la adiante a outros sujeitos do sistema d(a). É também possível, delegar somente a permissão de delegar determinada autorização. Neste caso, a delegação é chamada de delegação forte, pois não permite que o receptor tenha acesso à autorização em questão. Portanto, são possíveis a seguintes formas básicas de delegação:

- delegate(g, r, a): delegação simples de uma autorização.
- delegate(g, r, a+ud(a)):delegação de autorização mais o direito de delegá-la adiante.
- delegate(g, r, ud(a)): delegação forte de uma autorização.

É possível definir uma condição, representada por um predicado Q, para aceitação de determinada delegação. Além de definir condições para que determinada autorização seja delegada, é necessário em alguns casos definir se determinada autorização pode ser delegada adiante a terceiros, ou não. Caso seja possível delegá-la a terceiros, é interessante definir em até que profundidade isto é possível. Desta forma, além da condição, é possível definir o peso de uma delegação. Uma delegação incondicional de peso n é representada pela dupla ud(a, n), enquanto que uma condicional com peso n é representada por cd(a, Q, n). Uma delegação, condicional ou incondicional, com peso n é uma delegação que pode criar uma cadeia de delegações de no máximo n passos, tal que uma delegação no passo i suporta a delegação no passado i+1.

Dizemos que uma delegação d_1 = delegation(g_1, r_1, a_1+d(a_1)) suporta outra delegação d_2 = delegation(g_2, r_2, a_2+d(a_2)) se as seguintes condições são satisfeitas:

- r_1=g_2;
- o direito a_2 sendo delegado é no máximo tão forte quando o direito a_1;
- decrement(d(a_1))$\geq$ d(a_2), isto é, o direito de delegação em *d2* é no máximo tão forte quanto decrement(d(a_1)).

A relação de força entre duas autorizações, utilizada acima, é definida como T$\geq$ T' se e somente se:

- T = T' ou
- imply (T, T'), onde imply é a relação RBAC dois direitos.

A relação de força entre duas autorizações para delegação, por se tratar de autorizações especiais, é definida como:

- ud(T) $\geq$ ud(T,n) $\geq$ ud(T,k) for n > k ud(T) $\geq$ cd(T,Q), para todo Q
- ud(T,n) $\geq$ cd(T,Q,n), para todo n > 0 e Q
- cd(T,Q) $\geq$ cd(T,Q,n) $\geq$ cd(T,Q,k) se n > k
- Se T $\geq$ T' então cd(T,Q,n) $\geq$ cd(T',Q,n) e cd(T,Q) $\geq$ cd(T',Q)
- Se T $\geq$ T' então ud(T,n) $\geq$ ud(T',n),
- Se T $\geq$ T' então ud(T) $\geq$ ud(T'),
- Se Q e Q' são predicados simples, então cd(T,Q,n) $\geq$ cd(T,Q$\wedge$Q',n), para todo n e T.
- Se Q e Q' são predicados simples, então cd(T,Q) $\geq$ cd(T,Q$\wedge$Q') , para todo n e T.
- cd(T,Q,n) $\geq$ ud(T,0) para todo Q e n > 0.
- ud(T,0) $\geq$ ud(0) para todo T.

If P $\geq P_0$, dizemos que P_0 é mais fraca que P. Podemos também definir P > P_0 como P $\geq P_0$ e P $\neq P_0$, e chamamos > de relação estritamente forte. A função *decrement*, utilizada para representar a perda natural do poder de delegação ao longo de uma cadeia de delegações, possui as seguintes propriedades:

- decrement(ud(T,∞)) = ud(T,∞)
- decrement(ud(T,n)) = ud(T,n-1) para n > 0
- decrement(cd(T,Q,∞)) = cd(T,Q,∞)
- decrement(cd(T,Q,n)) = cd(T,Q,n-1) para n > 0

Finalmente, uma cadeia de delegações é representada por uma lista ordenada de delegações do tipo (G_i, D_i, R_i+d(R_i)), tal que:

- d_i suporta d_{i+1}
- nenhum g_i possui diretamente o direito R_i
- não há nenhum par d_i e d_j na cadeia, tal que $d_i = d_j$

Delegações são representadas através de um grafo simples dirigido com pesos, em que cada aresta representa uma delegação. Para cada objeto *o*, o grafo G[o] é utilizado para representar todas as delegações em relações ao objeto *o*. Seja G[o] = (V, E) um grafo dirigido com pesos, onde V é um conjunto finito de vértices representando os sujeitos que possuem alguma autorização em relação a *o*, E é um conjunto finito de arcos tal que se existe uma delegação (g, r, a), então (g, r) $\in$ E. Desta forma, o conjunto total de delegações G pode ser representado como G = {G[o] | o $\in O$}, ou seja, a união dos subgrafos relativos a cada objeto *o*. As definições apresentadas acima serão utilizadas a seguir na discussão do problema de revogação de delegações com downgrade (diminuição de peso).

4. Algoritmo para Revogação com Downgrade

Uma delegação inicial pode permitir a formação de uma cadeia independente de delegações de tamanho arbitrariamente grande, possuindo possivelmente ciclos. Ao revogarmos uma delegação que pertença a uma cadeia de suporte, para cada delegação d_i restante poderá ser necessário:

1. remover a delegação d_i pois a mesma não é mais alcançável a partir de outras cadeias de delegações, ou
2. diminuir o peso de d_i, para que esta delegação passe a possuir uma cadeia de suporte alternativa válida.

Estes dois casos são ilustrados na figura 1. Ao revogarmos a delegação de A a B, a delegação de B a C passa a não possuir uma cadeia de suporte, pois B não é alcançável a partir de qualquer outro sujeito. Portanto, a delegação de B a C deve ser removida. Isto ilustra o caso (1). As delegações CD, DE e EC devem ter seus pesos atualizados para que possam utilizar uma cadeia de suporte alternativa válida. No caso do sujeito D, que antes possuía a cadeia de suporte (AB, BC, CD), passa a possuir a cadeia de suporte alternativa (AE, EC, CD), ilustrando o caso (2). Entretanto, é necessário que os pesos ao longo da sequência de delegações (AE, EC, CD) sejam atualizados, para que a mesma possa ser considerada uma cadeia de suporte válida, com início no sujeito dono da autorização, representado por A.

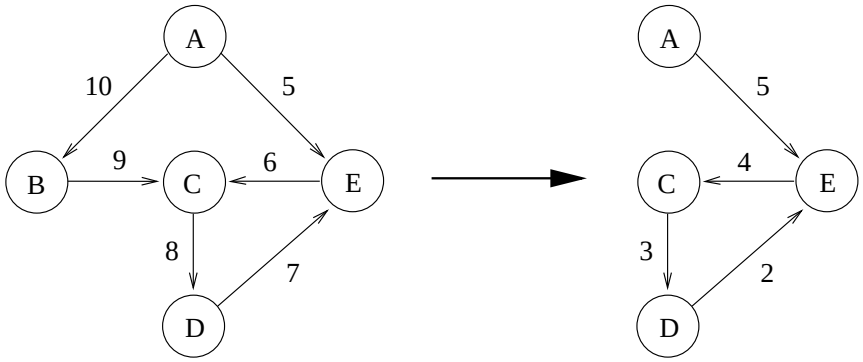

Figura 1. Revogação da delegação AB

Uma determinada delegação *d* possui uma cadeia de suporte válida se e somente se existe ao menos uma cadeia de delegações $d_0...d_k$ tal que d_k suporta *d*.

Definição 1: Um conjunto de delegações é válido se e somente se toda delegação possui ao menos uma cadeia de suporta válida.

Claramente, se um conjunto de delegações em relação a um único objeto e operação, representado pelo grafo G = (V, E), é válido, então toda aresta *uv* $\in$ G apresenta a seguinte propriedade:

Propriedade 1: w(uv) < max{w(s_iu)}, para todo s_iu $\in$ E.

Definição 2: Seja P = v_0...v um caminho em G. Dizemos que P é válido se para todo par de arestas subsequentes e_i e e_{i+1} em P, temos que w(e_{i+1}) < w(e_i).

Definição 3: o máximo peso acumulado em um vértice v_i, denotado por $\lambda(v_i)$, é o máximo peso que toda aresta v_iv_{i+1} pode adquirir, sem ferir a propriedade 1, e tal que o caminho $v_0...v_{i+1}$ seja um caminho válido.

Definição 4: o peso efetivo de uma aresta *uv*, denotado por $\psi(uv)$, é o mínimo valor entre w(uv) e $\lambda(u)$. Ou seja:

$\psi(uv)$ = min(w(uv),$\lambda(u)$)

Claramente, se G é válido, temos que w(uv) < $\lambda(u)$ para todo uv $\in$ E. Portanto a seguinte propriedade também é válida:

Propriedade 2: $\psi(uv)$ = w(uv), para todo uv $\in$ E.

Suponha que uma aresta tenha seu peso diminuído, tornando o grafo G inválido, ou seja, não obecendo as propriedades 1 e 2. Temos que recalcular os pesos de todas as arestas de G, de forma que estas propriedades sejam novamente satisfeitas, e que as delegações mantenham o máximo peso possível. Para tanto, é necessário para todo v $\in$ G, calcular $\lambda(v)$. Em seguida, é necessário recalcular os novos pesos de todas as arestas uv em G, denotados por w'(uv), utilizando-se a seguinte fórmula:

w'(uv) = $\psi(uv)$ = min(w(uv),$\lambda(u)$)

Assim garantimos que todas as arestas possuem o peso máximo permitido pelas cadeias alternativas, assim obedecendo as propriedades 1 e 2. O algoritmo de Dijkstra [Bondy and Murty 1976], ou uma variação deste, normalmente é utilizado para o cálculo do caminho mais curto de um vértice inicial a todos os outros vértices de um grafo. Entretanto, este algoritmo pode ser facilmente modificado para calcular o caminho mais longo, em lugar do caminho mais curto, de um vértice inicial u_0 aos outros vértices do grafo. Neste caso, u_0 representa o sujeito dono da autorização, e portando a raiz da cadeia de delegações.

Suponha que S é um subconjunto de V tal que $u_0 \in$ S, and seja $\overline{S}$ o conjunto complementar V\S. Se P = $u_0...\overline{uv}$ é o caminho mais longo de u_0 a $\overline{S}$ então claramente $\overline{u} \in$ S e a secção (u_0, $\overline{u}$) de P deve ser o caminho mais longo entre u_0 e $\overline{u}$. Portanto

d(u_0,$\overline{v}$) = d(u_0, $\overline{u}$) + w($\overline{uv}$)

E a distância mais longa de u_0 a $\overline{S}$ é dada pela fórmula:

d(u_0,$\overline{S}$) = $max_{u \in S, v \in \overline{S}}$\{d($u_0$, u) + w(uv)\}

Utilizando esta idéia, utilizaremos $\psi(uv)$ em lugar de d(u_0, u) + w(uv), para refletir o peso efetivo de cada aresta. Desta forma, obtemos:

d(u_0,$\overline{S}$) = $max_{u \in S, v \in \overline{S}}$\{$\psi(uv)$\}

Para o cálculo de $\psi(uv)$, é necessário o valor de $\lambda(u)$. À medida que o algoritmo apresentado a seguir prosegue, os vértices são rotulados de tal que forma que no estágio i, temos:

l(u) = d(u_0,u) para u $\in S_i$

l(v) = $max_{u \in S_{i-1}}$\{$\psi(uv)$\} para v $\in \overline{S}_i$

Podemos notar que l(u) é exatamente igual ao valor $\lambda(u)$, necessário para o cálculo de $\psi(uv)$ para toda aresta $uv \in E$.

Algoritmo **Downgrade**:

1. Seja l(u_0) = ∞, l(v) = -∞ para v $\neq u_0$, $S_0 = \{u_0\}$ e i = 0.
2. Para cada v $\in \overline{S_i}$, substitua l(v) por max{l(v), $\psi(u_i$v) - 1}. Calcule $max_{v \in \overline{S}_i}\{$l(v)$\}$ e seja u_{i+1} o vértice em que esse máximo é obtido. Faça $S_{i+1} = S_i \cup \{u_{i+1}\}$.
3. Se i = v - 1, pare. Se i < v -1, substitua i por i + 1 e volte para o passo 2.
4. Para cada $e \in$ E, substitua w(e) por $\psi(e)$, e se $\psi(e) < 0$, remova a aresta e.

Através da figura 2 são representados os caminhos, a partir do nó raiz até os outros nós, com maiores pesos acumulados (λ) antes (figura 2.a) e após (figura 2.c) a revogação da delegação AB. Na figura 2.b são apresentados os pesos acumulados em cada nó, que são obtidos, neste exemplo, após quatro iterações do passo 2. Ao término deste algoritmo temos os pesos de todas as arestas substituídos pelos seus respectivos pesos efetivos $\psi(e)$. Portanto, todas as arestas remanescentes pertencem a ao menos um caminho válido. A ordem de complexidade deste algoritmo é similar à do algoritmo de Dijkstra, citado acima. Para o algoritmo de Dijkstra são necessárias aproximadamente $\nu(\nu$-1)/2 adições e $\nu(\nu$-1) comparações. O algoritmo Downgrade, entranto, necessita de Δ(G) $\times$ $\nu(\nu$-1) comparações, devido ao cálculo de ψ, onde Δ(G) é máximo grau dos vértices em G. Desta forma, a complexidade do algortimo proposto é da ordem de $O(\Delta V^2)$. Este algoritmo foi implementado em Java para possível integração em um servidor de autorizações. Por motivo de concisão, o código fonte não será apresentado neste artigo.

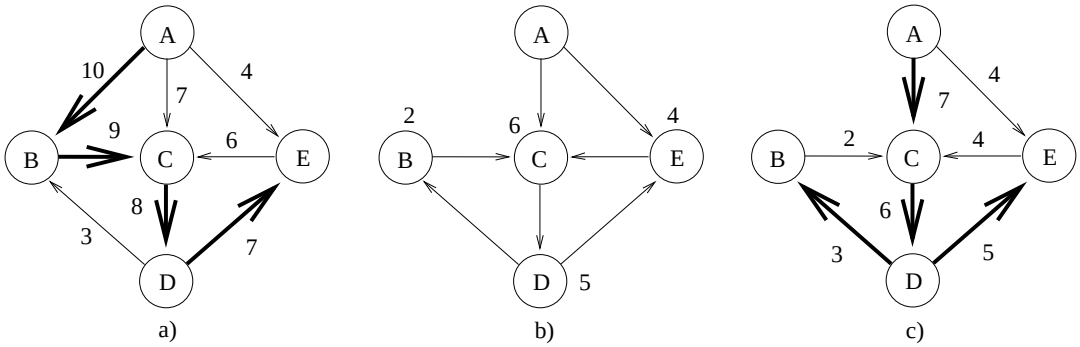

Figura 2. Caminhos com maiores pesos acumulados (λ)

5. Delegações Condicionais

Como citado anteriormente, é possível definir uma condição, representada por um predicado, ou expressão, simples Q, para aceitação de determinada delegação. Um exemplo de condição seria: ((Recipient.Department = Marketing) $\wedge$ (Recipient.Age $\geq$ 30)) $\vee$ (Recipient.Roles $\supset$ "Manager"). Neste exemplo, a delegação somente será aceita se o receptor pertencer ao departamento de Marketing e tiver idade maior ou igual a 30 anos, ou tiver o papel de gerente. Observe que é possível definir gama ampla de critérios de aceitação para delegações. Entretanto, a linguagem para especificação está fora do escopo deste artigo. Observe que para que uma determinada delegação seja aceita é satisfazer as condições impostas pelos sujeitos pertencentes à ao menos uma cadeia de suporte desta delegação. Anteriormente, com relação a delegações não condicionais, foi definida a seguinte propriedade fundamental para que um conjunto de delegações seja válido:

Propriedade 1: w(uv) < max{w($s_i u$)}, para todo siu em E.

Esta propriedade reflete o fato de cada delegação uv deve pertencer a ao menos uma cadeia de delegações tal que a delegação anterior possua um peso maior que o próprio peso de uv. Para delegações não condicionais, observe que esta propriedade é intuitivamente suficiente para garantir a validade de todas as delegações referentes à determinada autorização. Entretanto, com a introdução de delegações condicionais ao modelo, torna-se necessário redefinir o conceito de validade de uma cadeia de suporte. Seja $d_0...d_k$ uma cadeia de delegações, e sejam $Q_0...Q_k$ as condições associadas respectivamente a cada delegação pertencente a esta cadeia.

Definição 5: Uma delegação condicional d possui uma cadeia de suporte válida se e somente se existe ao menos uma cadeia de delegações $d_0...d_k$ tal que d_k suporta d, e tal que $Q_i(d_j)$ é verdadeiro para todo 0<i$\leq$k e todo 0<j<i.

A definição acima reflete o fato de que, ao longo de uma cadeia válida de delegações condicionais, o receptor de determinada delegação deve satisfazer todas as condições impostas às delegações anteriores pertencentes a esta cadeia. Tomando-se como base a nova definição de validade apresentada, é necessário estabelecer uma propriedade suficiente para garantir que um conjunto de delegações seja válido, assim como foi estabelecido para condições não condicionais utilizando-se a propriedade 1. Considere a delegação condicional entre dois sujeitos, denotada por d = delegate(g, r, a + cd(a,Q,w)). Sejam delegate(s_k, g, a + cd(a,Q_k,w_k)) as k delegações da autorização *a* tal que (s_kg) $\in$ E. Observe que a seguinte propriedade é valida:

Propriedade 3: $\bigcup_{i=0}^{k} (Q_i(d) \wedge (\text{w} < w_i))$

Entretanto, esta propriedade não é suficiente para garantir a validade de um conjunto de delegações, visto que a concordância com a condição imposta por uma delegação imediatamente anterior não implica que todas as condições ao longo de determinada cadeia sejam obedecidas, o que é necessário segundo a definição de validade apresentada acima. De fato, estamos interessados na condição efetiva imposta por determinada delegação, isto é, a condição ela própria impõe, somada às condições ao longo da cadeia de suporte desta delegação. Observe que, apesar de ser necessária somente uma cadeia de suporte válida para cada delegação, é possível a existência de mais de uma cadeia de suporte. Portanto, a condição efetiva imposta por determinada delegação deve refletir as condições impostas por todas as cadeias as quais esta delegação pertença. Não é possível, portanto, definir uma propriedade local, tal como no caso de delegações não condicionais, que seja suficiente para garantir a validade de um conjunto de delegações. A condição efetiva pode, entretanto, ser recursivamente definida como:

Q'= Q $\wedge$ ($\bigcup_{i=0}^{k} (Q'_i \wedge (\text{w} < w_i))$)

Com base na expressão para o cálculo da condição efetiva de delegações, podemos redefinir a propriedade 3:

Propriedade 4: $\bigcup_{i=0}^{k} (Q'_i \wedge (\text{w} < w_i))$

Esta propriedade, se atendida por todas as delegações, é suficiente para que um conjunto de delegações seja válido. Prova: Pela definição, um conjunto C de delegações em relação a uma autorização é válido se toda delegação d em C possui uma cadeia de suporte válida. Seja C um conjunto válido de delegações condicionais. Desta forma,

segundo a definição de condição efetiva, a propriedade 4 é válida para toda delegação em C. Suponha, por contradição, que ao acrescentarmos ao conjunto uma nova delegação d = delegate(g, r, a + cd(a,Q,w)), este passe a ser inválido, entretanto ainda obedecendo à propriedade 4. Sejam $t_0...t_k$ as delegações em C cujo receptor seja g. Pela proposição, as delegações $t_0...t_k$ possuem cada uma ao menos uma cadeia de suporte válida. Seja t_i a delegação tal que Q'_i(d) $\wedge$ (w $< w_i$) é verdadeiro, e seja $d_0...d_k$ a cadeia de suporte desta delegação. Então a cadeia $d_0, ..., d_k, t_i$ é uma cadeia de suporte válida para d, o que contraria a suposição de que C é inválido.

5.1. Aceitação de Delegações Condicionais

A introdução de delegações condicionais ao modelo, como explicado acima, impõe não somente uma condição à aceitação de cada nova delegação, mas sim um número de condições igual ao da cadeia de suporte desta delegação. Intuitivamente, quando mais distante uma delegação se encontra do dono da autorização, mais condições o sujeito receptor de determinada delegação deve satisfazer. Para determinar se determinada delegação pode ser concedida, devemos verificar se a mesma possui uma cadeia de suporte válida, tanto em relação ao seu peso, quanto em relação às condições impostas ao longo da cadeia de suporte.

Para uma mesma delegação, pode ser possível a existência de uma ou mais cadeias de suporte válidas, entretanto estamos preocupados em estabelecer um algoritmo para determinar se nova delegação possui ao menos uma destas cadeias. Ou seja, devemos garantir que a propriedade 4, anteriormente apresentada, é válida para a nova delegação, já que esta propriedade é suficiente para garantir a validade do conjunto de delegações. O algoritmo proposto a seguir utiliza a própria definição de condição efetiva para determinar se existe uma cadeia de suporte para a nova delegação.

Algoritmo para Aceitação de Delegações:

1. Seja e=u_0v_0 a delegação a ser acrescentada ao conjunto de delegações, e r_0 o vértice raiz do grafo. Faça C = $\{u_0\}$, W = $\overline{C}$
2. Faça C = C + {u} para todo uv, tal que $Q_{uv}(u_0v_0)$ é satisfeita, u $\in$ W, v $\in$ C. Faça W = W - {u} e C = C - {v}.
3. Se C $\supset r_0$, então o grafo é válido. Se W = $\emptyset$, o grafo é inválido. Senão repita o passo 2.

A partir da delegação acrescentada, este algoritmo tenta encontrar um caminho até o nó raiz, tal que ao longo deste caminho a propriedade 3 seja satisfeita. Observe que $Q_{uv}(u_0v_0)$ é sempre avaliada em relação a u_0v_0, isto é, verifica-se se a condição Q_{uv} definida pela delegação uv é satisfeita pela delegação u_0v_0. A complexidade deste algoritmo é O(E), visto que cada aresta é visitada no máximo uma única vez. Observe que o algoritmo verifica se existe ao menos uma cadeia de suporte válida, mas não indica qual seja esta cadeia, ou se a existência de cadeias de suporte alternativas é possível.

5.2. Revogação de Delegações condicionais

Ao contrário do que ocorre em um conjunto formado inteiramente por delegações não condicionais, o efeito da revogação de uma delegação em um conjunto de delegações condicionais depende não somente do peso desta delegação, mas também da condição por

ela imposta. Se determinada delegação não pertencer a uma cadeia de suporte de outras delegações, sua revogação não terá efeito algum, a não ser sobre o próprio sujeito que a recebeu, que possivelmente perderá o direito sobre a autorização anteriormente delegada. Por outro, caso esta delegação pertença a uma cadeia de suporte, possivelmente outras delegações deverão ter seus pesos reduzidos para poderem utilizar cadeias de suporte alternativas.

Com base no algoritmo para revogação com downgrade anteriormente proposto, obtemos uma forma eficiente de se manter consistente o conjunto de autorizações referentes a uma determinada autorização. Neste caso, entretanto, considerou-se a validade de cadeias de delegações somente com base nos pesos máximos permitidos para cada delegação. Agora, ao introduzirmos o conceito de delegações condicionais ao modelo, para determinar a validade de uma cadeia de delegações, torna-se também necessário satisfazer as condições impostas ao longo desta cadeia, como discutido acima. A introdução do conceito de delegações condicionais tem conseqüências, portanto, no algoritmo proposto para revogação com downgrade, na medida em que uma cadeia de suporte válida formada por delegações não condicionais pode passar a ser inválida se considerarmos a concordância ou não das condições ao longo desta cadeia em relação à delegação em questão. O algoritmo para revogação com downgrade para delegações não condicionais utiliza o conceito do peso efetivo de cada aresta: $\psi(uv)$ = min(w(uv),$\lambda(u)$)

O peso efetivo foi utilizado para recalcular o peso de cada delegação levando-se em consideração o máximo peso permitido por sua cadeia de suporte. Observe que ψ não pode ser diretamente utilizado no caso de delegações condicionais, visto que $\lambda(u)$, o máximo peso acumulado em um vértice, depende da aresta de destino. A principal conseqüência desta observação é que temos que considerar que o peso a ser transferido de uma delegação à outra imediatamente adjacente pode ser menor que o peso da primeira delegação subtraído de uma unidade. Isto se deve ao fato de que duas delegações adjacentes podem possuir cadeias de suporte diferentes, mesmo que a primeira delegação pertença à cadeia de suporte da segunda. Ou seja, a cadeia que permitiu determinado peso à primeira delegação, pode não ser válida como suporte para a segunda delegação. Portanto, neste caso, é possível que segunda delegação tenha que utilizar uma cadeia de suporte que permita um peso inferior ao que seria possível se utilizasse a mesma cadeia de suporte da primeira delegação. Ou seja, o peso da primeira delegação, digamos w, impõe restrições ao peso da segunda delegação, neste caso no máximo w-1, mas não é suficiente para garantir que o peso da segunda delegação seja realmente w-1.

Um exemplo desta situação é apresentado na figura 3. Para tornar o entendimento mais simples, foi representado somente um subgrafo do conjunto total de delegações em relação a uma determinada autorização. Considere que delegações AC, BC e CD possuem as condições Q_{ac}, Q_{bc} e Q_{cd}, respectivamente. Suponha que estas três condições são atendidas pela delegação DE, mas somente as condições Q_{ac} e Q_{cd} são atendidas por DF. Observa-se que o máximo peso acumulado em D em relação à DE é 6, enquanto que o máximo peso acumulado neste mesmo vértice em relação à DF é 3. Isto ocorre porque a delegação DE pode utilizar a cadeia de suporte (BC, CD), enquanto que delegação DF deve utilizar a cadeia de suporte alternativa (AC, CD), visto que a condição imposta por Q_{bc} não é atendida por DF. Observe que se este mesmo subgrafo fosse formado por delegações não condicionais, o máximo peso acumulado em D seria o mesmo para DE e

DF, ou seja, 6.

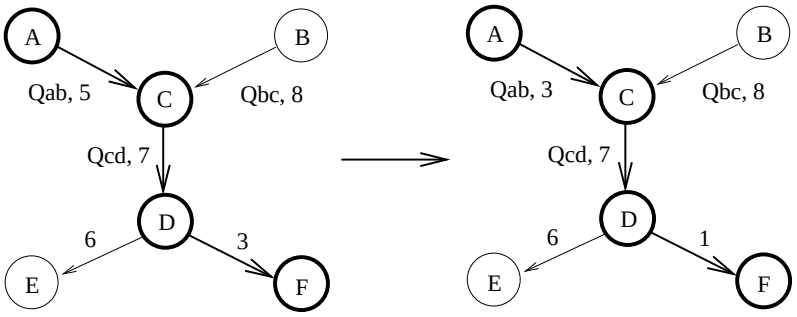

Figura 3. Cadeia de Suporte da Delegação DF

5.3. Algoritmo Downgrade Condicional

A seguir será apresentado um primeiro algoritmo de downgrade para delegação condicionais, considerando as questões levantadas acima. A solução mais simples para um algoritmo de downgrade condicional, é utilizar a mesma idéia do algoritmo para delegações não condicionais. Entretanto, para tanto, é preciso recalcular λ(u), necessário para o cálculo de ψ(uv) no passo 2, dependendo da aresta uv em questão. Denotemos este novo valor de λ(u), por λ_{uv}(u), representado o máximo peso acumulado em u em relação à aresta uv. A definição de ψ pode ser reformulada como:

$\psi'(uv)$ = min(w(uv),$\lambda_{uv}(u)$)

Observe que λ_{uv} reflete o fato de que as arestas cuja condição não seja atendida por uv possuem um peso efetivo nulo em relação a uv. Isto é, tais arestas não podem pertencer a uma cadeia de suporte de uv. Para obtermos λ_{uv}, podemos, primeiramente, ajustar os pesos das arestas para refletir esta propriedade. Denotemos por w_{uv}(e) o peso da aresta *e* em relação à aresta uv. Este peso é obtido aplicando-se a seguinte a toda aresta e em E:

- w_{uv}(e) = w(e) se Q_e(uv) = true
- w_{uv}(e) = 0 caso contrário.

Denotemos por G_{uv} o grafo obtido após a aplicação desta regra. Observe que, se obtermos o valor de λ(u) neste novo grafo, este valor será justamente o valor que queremos obter, ou seja, λ_{uv}(u). Podemos obtê-lo aplicando o algoritmo para delegações não condicionais em G_{uv}. Com este modificação, a complexidade do algoritmo também é altamente alterada, uma vez que em cada iteração do passo 2, devemos aplicar uma instância do algoritmo downgrade não condicional ao subgrafo induzido pelo conjunto S_i. Assim, a complexidade deste algoritmo inicialmente proposto é da ordem de O(V^4).

Algoritmo **Downgrade Condicional**:

1. Seja l(u_0) = ∞, l(v) = -∞ para v $\neq u_0$, S_0 = {u_0} e i = 0.
2. Para cada v $\in \overline{S_i}$, aplique *Downgrade* ($G_{u_i v}[S_i]$), e substitua l(v) por max{l(v), $\psi'(u_i$v) - 1}. Calcule $max_{v \in \overline{S}_i}${l(v)} e seja u_{i+1} o vértice em que esse máximo é obtido. Faça $S_{i+1} = S_i \cup \{u_{i+1}\}$.
3. Se i = *v* - 1, pare. Se i < *v* -1, substitua i por i + 1 e volte para o passo 2.
4. Para cada *e* $\in$ E, substitua w(*e*) por ψ(*e*), e se ψ(*e*) < 0, remova a aresta *e*.

6. Trabalho Futuro

A seguir serão destacados os principais pontos que ainda devem ser refinados em um trabalho futuro, visando tornar o modelo proposto mais amplo.

6.1. Suporte a autorizações negativas

Atualmente o modelo não suporta o conceito de autorizações negativas. Da mesma forma que é possível a um sujeito delegar autorizações a outros sujeitos, o modelo deve permitir a este sujeito proibir que outros sujeitos possuam determinada autorização, mesmo que a mesma já tenha sido concedida por outro sujeito. O emprego de autorizações negativas é especialmente útil para a definição de exceções, quando autorizações são concedidas a grupos de sujeitos. A introdução desse problema deve levar a alterações nos algoritmos de aceitação e revogação de delegações. Deve ser estabelecido um modelo formal que permita definir políticas para resolução de conflitos, quando determinado sujeito possuir autorizações positivas e negativas em relação ao mesmo objeto.

6.2. Aplicando Restrições MAC ao Modelo

De uma maneira geral, pode-se dizer que as políticas DAC e MAC não são mutuamente exclusivas, pois podem ser utilizadas em conjunto. Neste caso, para que o acesso a determinado objeto seja permitido, é necessário que o sujeito possua autorização para isso, e que as condições impostas pelo MAC sejam satisfeitas. Ou seja, em tal situação a política DAC atua dentro dos limites impostos pela política MAC, uma vez que a política DAC pode restringir somente os acessos que são permitidos pela política MAC. Segundo o modelo Bell e LaPadula [Samarati and di Vimercati 2001] um sistema é composto de um conjunto de sujeitos *S*, objetos *O*, e ações *A*, as quais incluem *read* e *write*. O modelo também assume que exista um conjunto *L* de classes de acesso e uma função λ: $S \cup O$ -> *L*, que quando aplicada a um sujeito ou objeto retorna a sua classificação neste estado. Segundo a propriedade simples definida pelo modelo, um sujeito *s* possui acesso de leitura (*read*) a determinado objeto *o* somente se $\lambda(s) \geq \lambda(o)$, e possui acesso de escrita (*write*) somente se $\lambda(o) \geq \lambda(s)$. Restrições MAC podem ser aplicadas no modelo proposto através de delegações condicionais. O modelo Bell e LaPadula pode ser mapeado para operações de leitura pela condição (Recipient.Clearance $\geq$ Autorization.Object.AccessClass), e para operações de escrita pela condição (Autorization.Object.AccessClass $\geq$ Recipient.Clearance). Desta forma, as duas propriedades principais do modelo Bell LaPadula são satisfeitas. Entretanto, assim como a função λ reflete mudanças de estado, a avaliação de condições também deve ser dependente do estado do sistema, refletindo possíveis alterações em relação às classes de acesso de sujeitos e objetos.

7. Conclusão

Neste documento foi apresentado um modelo que tem como principal vantagem, em relação a outros trabalhos relacionados à área de modelos de autorização, a capacidade de controlar de forma precisa a formação de cadeias de delegações. Como apresentado, os dois conceitos que permitem tal controle são o conceito de comprimento de cadeias de delegações, e o conceito de delegações condicionais.

Entre as principais contribuições deste documento, estão a definição de algoritmos eficientes, isto é polinomiais, de revogação, bem como de aceitação, para o modelo apresentado, caracterizando-se por manter os sujeitos remanescentes do sistema com o maior

conjunto possível de privilégios após a revogação de delegações. Finalmente, como trabalho futuro, foram discutidas possíveis extensões ao modelo proposto, especialmente autorizações negativas, permitindo um controle ainda melhor sobre delegações.

Referências

[A.K. Jones and Snyder. 1976] A.K. Jones, R. L. and Snyder., L. (1976). A linear time algorithm for deciding security. In *Proc. 17th Anual Symposium on Foundations of Computer Science*, pages 33–41.

[Astrahan 1976] Astrahan, M. M. (1976). System R: A relational approach to database management. *ACM Trans. Database Syst.*, 1(2):97–137.

[Barka and Sandhu. 2000] Barka, E. and Sandhu., R. (2000). A role-based delegation model and some extensions. In *23rd National Information Systems Security Conference, Baltimore, MD, October 2000.*

[Bertino et al. 1997] Bertino, E., Samarati, P., and Jajodia, S. (1997). An extended authorization model for relational databases. *Knowledge and Data Engineering*, 9(1):85–101.

[Bondy and Murty 1976] Bondy, J. A. and Murty, U. S. R. (1976). *Graph Theory with Applications*. American Elsevier.

[Griffiths and Wade 1976] Griffiths, P. P. and Wade, B. W. (1976). An authorization mechanism for a relational data base system (abstract). In Rothnie, J. B., editor, *SIGMOD Conference*, page 51. ACM.

[Ruan and Varadharajan 2004] Ruan, C. and Varadharajan, V. (2004). A weighted graph approach to authorization delegation and conflict resolution. In Wang, H., Pieprzyk, J., and Varadharajan, V., editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 402–413. Springer.

[Samarati and di Vimercati 2001] Samarati, P. and di Vimercati, S. D. C. (2001). Access control: Policies, models, and mechanisms. In Focardi, R. and Gorrieri, R., editors, *Foundations of Security Analysis and Design*, LNCS 2171. Springer-Verlag.

[U. Bussolati and Martella. 1993] U. Bussolati, M. F. and Martella., G. (1993). A conceptual framework for security system: the action-entity model. In *Proc. 9th IFIP World Conference*, pages 127–132.

[Wainer et al. 2003] Wainer, J., Barthelmess, P., and Kumar, A. (2003). W-RBAC - A workflow security model incorporating controlled overriding of constraints. *Int. J. Cooperative Inf. Syst.*, 12(4):455–485.

[Wainer et al. 2005] Wainer, J., Kumar, A., and Barthelmess, P. (2005). DW-RBAC: A formal security model of delegation and revocation in workflow systems. *Information Systems*, To be published.

[Woo and Lam 1993] Woo, T. Y. C. and Lam, S. S. (1993). Authorizations in distributed systems: A new approach. *Journal of Computer Security*, 2(2-3):107–136.

[Yao et al. 2001] Yao, W., Moody, K., and Bacon, J. (2001). A model of oasis role-based access control and its support for active security. In *SACMAT*, pages 171–181.

[Zhang et al. 2003a] Zhang, L., Ahn, G.-J., and tseng Chu, B. (2003a). A rule-based framework for role-based delegation and revocation. *ACM Trans. Inf. Syst. Secur.*, 6(3):404–441.

[Zhang et al. 2003b] Zhang, X., Oh, S., and Sandhu., R. (2003b). Pbdm: A flexible delegation model in rbac. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies.*