IT-Sicherheitsmanagement Seminar: Group-Centric Models for Secure Information Sharing (g-SIS)

Uwe Kühn, Tobias Pöppke

1 Einleitung

TODO: Subscription Service, Meeting room Metaphern. [KSNW09] [SKNW10]

2 Grundlagen

Policy-, Enforcement-Layer.

2.1 Linear Temporal Logic

2.2 Lattice-Based Access Control

Eine wichtige Klasse von Zugriffskontrollmodellen sind *Lattice-Based Access Control (LBAC)* Modelle. Die Grundlagen von LBAC wurden in den 1970ern von Bell, LaPadula, Biba und Denning gelegt und entstanden aus den Bedürfnissen des Verteidigungssektors. Eine Übersicht über die verschiedenen Modelle und formale Definitionen, sowie weiterführende Referenzen finden sich bei Sandhu [Sand93].

LBAC Modelle konzentrieren sich auf die Betrachtung des Informationsflusses zwischen Sicherheitsklassen eines Systems. Dazu wird jedem *Objekt* des Systems eine *Sicherheitsklasse* zugewiesen. Ein *Objekt* sei hier informell definiert als ein Container für Informationen. Wenn nun die Informationen eines Objekts x zu einem Objekt y fließen geht damit ein Informationsfluss einher. Dieser Informationsfluss findet zwischen den Sicherheitsklassen der jeweiligen Objekte statt.

Die *Richtlinien* mit denen die Informationsflüsse beschrieben werden, können mithilfe von *Lattices* beschrieben werden. Ein *Lattice* ist in diesem Zusammenhang die partiell geordnete Menge der Sicherheitsklassen. Jedem Objekt wird eine Sicherheitsklasse zugewiesen. Im Modell von Bell und LaPadula gelten dann Beispielsweise die folgenden Aussagen. Benutzer können *Subjekte*, wie zum Beispiel einen Prozess, erstellen, die eine Sicherheitsklasse besitzen, die von der Sicherheitsklasse des Benutzers im Lattice dominiert wird. Ein Subjekt kann genau dann Objekte lesen, wenn die Sicherheitsklasse des Objekts von der des Subjekts dominiert wird. Auf ein Objekt schreibend zugreifen kann ein Subjekt genau dann, wenn die Sicherheitsklasse des Objekts die Sicherheitsklasse des Subjekts dominiert.

2.3 Domain and Type Enforcement

Die Idee des *Domain and Type Enforcement (DTE)*, von Badger et al. vorgeschlagen [BSSW+95], ist eine Erweiterung der Lattice-Based Access Control um *Domänen* und *Typen*. Eine *Domäne*

enthält Subjekte und Objekte werden zu *Typen* zugeordnet. Um den Informationsfluss zu kontrollieren, wird eine Matrix benutzt, in der die Lese- und Schreibrechte für jede Kombination von Domäne und Typ festgelegt werden. Diese Art der Zugriffskontrolle ist beispielsweise dann anwendbar, wenn es gewünscht ist, vertrauenswürdige Pipelines zu etablieren. Wenn zum Beispiel gewünscht wird, dass Informationen nur über ein Subjekt einer dritten Domäne zum Zielobjekt fließen dürfen, ist dies mit klassischen LBAC-Modellen nicht möglich. Dies liegt an der transitiven Natur der zugrundeliegenden Relation in LBAC.

Da die im Lattice höher angesiedelten Sicherheitsklassen in LBAC alle unter ihnen liegenden Sicherheitsklassen dominieren und diese Relation transitiv ist, kann Information immer von allen dominierten Sicherheitsklassen direkt in die dominierende Sicherheitsklasse fließen. Da bei DTE eine Matrix den zulässigen Informationsfluss definiert, ist diese Relation nicht länger transitiv und die Informationen können nach Wunsch über verschiedene Wege fließen.

2.4 Role-Based Access Control

Role-Based Access Control (RBAC) ist ein Zugriffskontrollmodel, dass von Sandhu et al. und Ferraiolo et al. vorgeschlagen wurde [SCFY96, FSGK$^+$01]. Ein Benutzer sei im Folgenden, ohne Beschränkung der Allgemeinheit, definiert als ein menschlicher Benutzer. Das Hauptmerkmal von RBAC sind *Rollen*. Eine Rolle basiert auf der Idee eine Position innerhalb einer Organisation darzustellen, die mit bestimmten Rechten und Verantwortlichkeiten einhergeht. Daher werden jeder Rolle bestimmte *Zugriffsrechte* auf *Objekte* zugewiesen, wobei ein *Objekt* Informationen bereitstellen oder erhalten kann. Ein *Zugriffsrecht* gewährt einer Rolle die Möglichkeit eine oder mehrere *Operationen* auf einem oder mehreren Objekten auszuführen.

Ein Benutzer kann in RBAC mehreren Rollen zugehören und eine Rolle kann ebenfalls mehreren Benutzern zugeordnet sein. Diese Relation wird als *User Assignment (UA)* bezeichnet. Genau so können auch Zugriffsrechte zu Rollen in einer many-to-many Relation zugewiesen werden, die als *Permission Assignment (PA)* bezeichnet wird.

Ein weiteres Konzept das in RBAC benutzt wird, sind *Sessions*. Ein Benutzer kann einer oder mehreren Sessions zugeordnet sein und jede Session ist genau einem Benutzer zugeordnet. In jeder Session kann wiederum eine Untermenge der Rollen, denen der Benutzer angehört, aktiviert sein. Welche Rollen, und damit auch welche Zugriffsrechte, in einer Session aktiviert sind, kann durch die Funktion *session_roles* abgefragt werden. Um zu erfahren, welche Sessions zu einem Benutzer gehören, kann die Funktion *user_sessions* aufgerufen werden.

Die obigen Eigenschaften beschreiben *Core RBAC* beziehungsweise $RBAC_0$. Formal kann damit Core RBAC wie folgt definiert werden.

Definition 1 (Core RBAC)

- USERS, ROLES, OPS und OBS*: Mengen der Benutzer, Rollen, Operationen und Objekten.*
- $UA \subseteq USERS \times ROLES$*: Menge der Zuweisungen von Benutzern zu Rollen.*
- $assigned\_users : (r : ROLES) \rightarrow 2^{USERS}, assigned\_users(r) = \{u \in USERS | (u, r) \in UA\}$*: Zuordnung der Rolle* r *zu der zugehörigen Menge der Benutzer.*
- $PRMS = 2^{(OPS \times OBS)}$*: Menge der Zugriffsrechte.*
- $PA \subseteq PRMS \times ROLES$*: Zuordnung der Zugriffsrechte zu Rollen.*
- $assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}, assigned\_permissions(r) = \{u \in USERS | (u, r) \in PA\}$*: Zuordnung der Rolle* r *zu der zugehörigen Menge der Zugriffsrechte.*

- $Ob(p : PRMS) \rightarrow \{op \subseteq OBS\}$*: Gibt die Menge der Operationen an, die einem Zugriffsrecht* p *zugeordnet sind.*
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$*: Gibt die Menge der Objekte an, die einem Zugriffsrecht* p *zugeordnet sind.*
- SESSIONS*: Menge der Sessions.*
- $user\_sessions(u : Users) \rightarrow 2^{SESSIONS}$*: Zuordnung eines Benutzers* u *zu einer Menge von Sessions.*
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}, session\_roles(s_i) \subseteq \{r \in ROLES | (session\_users(si), r) \in UA\}$*: Zuordnung einer Session* s *zu der zugehörigen Menge von Rollen.*
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}, \bigcup r \in session\_roles(s) assigned\_permissions(r)$*: Die Zugriffsrechte, die einem Benutzer in einer Session zur Verfügung stehen.*

Das Core RBAC Modell definiert die grundlegenden Eigenschaften jedes RBAC Modells. Es wurden auch weitergehende Modelle beschrieben, wie hierarchisches RBAC und constrained RBAC. Für die Definitionen dieser Modelle sei an dieser Stelle jedoch lediglich auf Ferraiolo et al. [FSGK+01] verwiesen

TODO: Übergang zum nächsten Abschnitt

3 G-SIS

3.1 Klassifizierung von g-SIS Modellen

In vielen Fällen ist es notwendig, die Benutzer eines abzusichernden Systems in mehr als eine Gruppe einzuteilen. Diese Gruppen können sowohl nach der Art der Relationen zwischen den einzelnen Gruppen unterschieden werden, als auch nach der Art der Beziehungen zwischen den Benutzern innerhalb einer Gruppe.

Verschiedene Gruppen in g-SIS könenn *verbunden* oder *isoliert* sein. Sind Gruppen untereinander *isoliert*, hat die Mitgliedschaft eines Benutzers, Subjekts oder Objekts keinen Einfluss auf die Mitgliedschaft eines Benutzers, Subjekts oder Objekts in einer anderen Gruppe.

Im Falle mehrerer *verbundener* Gruppen kann die Mitgliedschaft in einer Gruppe verschiedene Auswirkungen auf die Mitgliedschaft und die möglichen Operationen in einer anderen Gruppe haben.

Die Mitglieder innerhalb einer Gruppe können wiederum alle die gleichen Zugriffsrechte besitzen. Diese Art der Gruppe wird in g-SIS definiert als eine *undifferenzierte* Gruppe. Im Gegensatz dazu können in einer *differenzierten* Gruppe verschiedenen Benutzern auch verschiedene Zugriffsrechte eingeräumt werden.

Die genannten Unterschiede und ihre Kombinationen führen zu vier möglichen g-SIS Modellen. Diese sind das *isoliert undifferenzierte Modell*, das *isoliert differenzierte Modell*, das *verbunden undifferenzierte Modell* sowie das *verbunden differenzierte Modell*.

Das *isoliert undifferenzierte Modell* beschreibt das Modell, in dem die Mitgliedschaft in einer Gruppe keinen Einfluss auf andere Gruppen im System hat. Ebenso gibt es innerhalb der einzelnen Gruppen keinen Unterschied in den Zugriffsrechten der einzelnen Mitglieder, außer den Unterschieden, die durch die g-SIS Core Properties (s.h. Abschnitt **??**) für die Zugriffsrechte festgelegt werden. Das isolierte undifferenzierte Modell bildet damit die Grundlage für die anderen g-SIS Modelle.

Im *verbundenen undifferenzierten Modell* (im Folgenden g-SIS^c) kann die Mitgliedschaft in einer Gruppe verschiedene Auswirkungen auf andere Gruppen haben, die im folgenden Abschnitt genauer betrachtet werden. Desweiteren haben die Mitglieder untereinander, wie im isolierten undifferenzierten Modell, die selben Zugriffsrechte. In Abbildung 1 sind die Beziehungen zwischen den einzelnen g-SIS Modellen grafisch dargestellt. Das isoliert undifferenzierte Modell ist in allen anderen Modellen enthalten. Das isoliert differenzierte Modell und das verbunden undifferenzierte Modell sind in diesem Sinne nicht vergleichbar. Das verbunden differenzierte Modell wiederum enthält alle anderen Modelle.

Im Folgenden soll allerdings lediglich das verbunden undifferenzierte Modell betrachtet werden, da dies ausreicht um Aussagen über die Relationen zwischen Gruppen machen zu können.

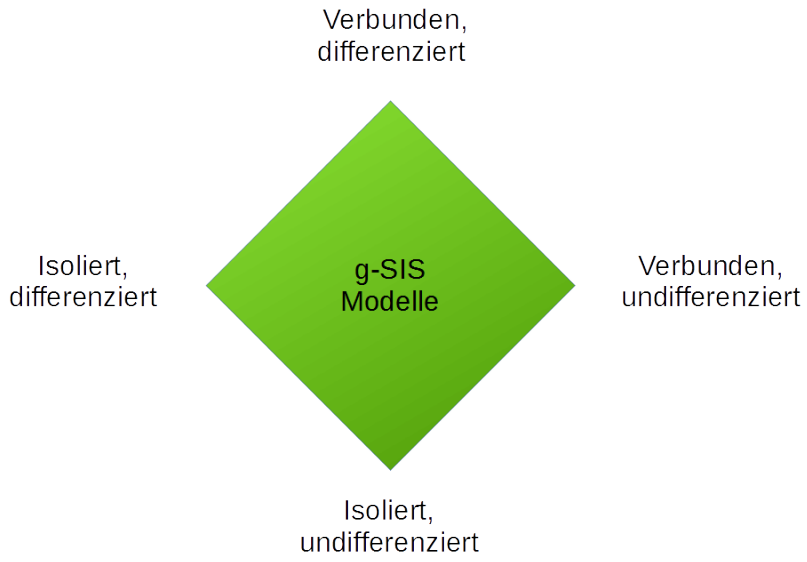

Figure 1: Klassifikation der verschiedenen g-SIS Modelle

3.2 Relationen zwischen Gruppen

Im verbunden undifferenzierten g-SIS Modell werden verschiedene Relationen für die Verbindungen zwischen einzelnen Gruppen vorgeschlagen. Die betrachteten Entitäten sind wiederum Benutzer, denen ein gewisses Vertrauen entgegengebracht wird, sowie Subjekte und Objekte. Subjekte, wie Prozesse, die von einem Benutzer erstellt werden, müssen nicht notwendig die selben Rechte besitzen wie der Benutzer, der sie erstellt hat.

Die folgenden Relationen zwischen Gruppen wurden vorgeschlagen:

1. Conditional Membership (condM): Diese Relation definiert, dass die Mitgliedschaft eines Benutzers in einer Gruppe abhängig von seiner Mitgliedschaft in einer anderen Gruppe ist. Diese Relation ist als reflexiv, aber nicht als transitiv oder symmetrisch definiert und macht lediglich Aussagen über Gruppen für Benutzer, nicht für Subjekte. Ist Transitivität oder Symmetrie gewünscht, so muss dies explizit definiert werden. In der Konfiguration aus Abbildung 2 wird zum Beispiel durch die Definition von condM(Meeting, {Manager, Experten 1, Experten 2}) verlangt, dass ein Benutzer nur Mitglied in der

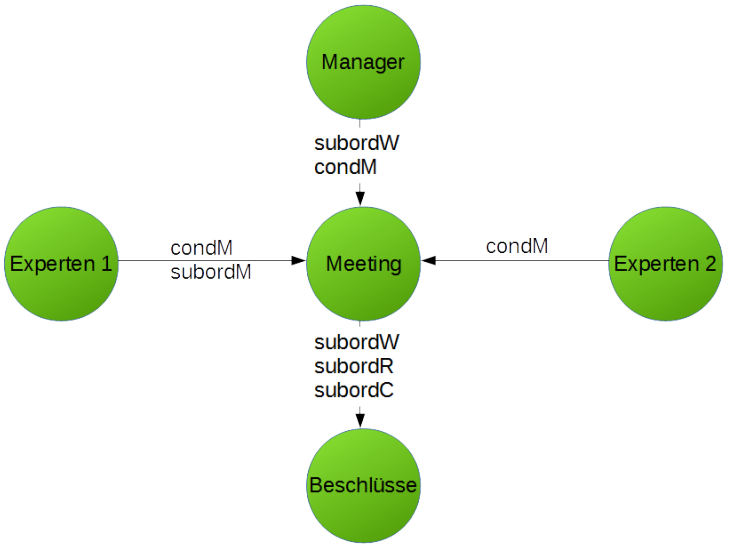

Figure 2: Ein Beispiel für die Relationen zwischen den Gruppen eines Meetings

Meeting-Gruppe sein kann, wenn er Mitglied in einer der Gruppen Manager, Experten 1 oder Experten 2 ist.

2. Subordination: Diese Relationen definieren die Dominanz einer Gruppe über eine andere in verschiedenen Aspekten. Wie die Conditional Membership ist auch diese Relation reflexiv, aber nicht transitiv oder symmetrisch.

   - Create Subordination (subordC): Benutzer aus der dominierenden Gruppe können Subjekte in der dominierten Gruppe erstellen. Die Definition von subordC(Meeting, Beschlüsse) erlaubt das Erstellen von Subjekten in der Gruppe Beschlüsse durch Benutzer aus Meeting.
   - Read Subordination (subordR): Subjekte aus der dominierenden Gruppe können Objekte in der dominierten Gruppe lesen. Durch subordR(Meeting, Beschlüsse) können Subjekte in Meeting auf Objekte in Beschlüsse lesend zugreifen.
   - Write Subordination (subordW): Subjekte aus der dominierenden Gruppe können auf Objekte in der dominierten Gruppe schreibend zugreifen. Aufgrund von subordW(Manager, Meeting) kann ein Subjekt in der Manager Gruppe auf Objekte in der Meeting Gruppe schreibend zugreifen.
   - Move Subordination (subordM): Subjekte können von der dominierenden Gruppe in die dominierte Gruppe verschoben werden. Mit subordM(Experten 1, Meeting) kann ein Subjekt aus Experten 1 in die Meeting Gruppe verschoben werden, kann dadurch aber den Zugriff auf Objekte aus der Experten 1 Gruppe verlieren.

3. Mutual Exclusion: Zwei Gruppen können als sich gegenseitig ausschließend definiert werden. Das führt dazu, dass Benutzer oder Objekte nicht zur gleichen Zeit in Mitglied in beiden Gruppen sein dürfen. Es ist jedoch möglich dieses Kriterium als dynamisch zu definieren, wodurch es möglich wird, dass ein Benutzer zwar Mitglied in beiden Gruppen ist, jedoch nicht in beiden Gruppen zur selben Zeit Subjekte erstellen kann.

4. Kardinalität: Eine Vielzahl an Beschränkungen der Kardinalität, wie zum Beispiel eine Kardinalität für die Anzahl der Benutzer, Subjekte oder Objekte in einer Gruppe, können definiert werden.

In Abbildung 2 ist eine Beispielkonfiguration von g-SIS^c zu sehen. Die Relationen innerhalb eines Systems können sich in g-SIS^c auch mit der Zeit ändern, wenn sich die Anforderungen an das System ändern. Dies ist ein wichtiger Aspekt von g-SIS^c, da das Teilen von Informationen in einem System von den momentanen Bedürfnissen der Benutzer abhängt.

4 Andere Zugriffskontrollmechanismen in g-SIS^c

Mit Hilfe der beschriebenen Eigenschaften der g-SIS Modelle ist es nun möglich, bekannte Zugriffskontrollmodelle mit g-SIS zu modellieren. Wie weiter oben schon erwähnt, reicht es hier und im Folgenden aus, das verbunden undifferenzierte Modell für diese Untersuchung zu betrachten.

4.1 Lattice-Based Access Control in g-SIS^c

In g-SIS^c können LBAC Richtlinien, wie zum Beispiel das Bell-LaPadula Modell, modelliert werden. Um ein bestehendes Bell-LaPadula System in g-SIS^c zu konstruieren, muss für jede Sicherheitsklasse in LBAC eine Gruppe in g-SIS^c erstellt werden. Für jede Sicherheitsklasse H, die eine Sicherheitsklasse L im Sinne von LBAC dominiert, müssen für die zugehörigen Gruppen G_H und G_L die Relationen subordR(G_L, G_H), subordC(G_L, G_H) und subordW(G_H, G_L). Da die Relationen in LBAC jedoch transitiv sind, muss darauf geachtet werden, diesen Schritt auch mit jeder Sicherheitsklasse zu wiederholen, die L dominiert. Für ein Beispiel dieser Transformation sei hier auf Sandhu et al. [SKNW10] verwiesen.

Für den Fall, dass zwei Organisationen mit unterschiedlichen LBAC Lattices Informationen austauschen wollen, kann dies im klassischen LBAC nicht durch einfache Anpassungen an den Lattices bewerkstelligt werden. Wurden die Richtlinien jedoch mit g-SIS^c modelliert, ist es beispielsweise möglich alle Objekte mit Sicherheitsklasse L in Organisation A für alle Benutzer in Organisation B mit Sicherheitsklasse TS freizugeben. Dazu muss lediglich die Relation subordR(G_B_TS, G_A_L) definiert werden.

4.2 Domain and Type Enforcement in g-SIS^c

Wie in Abschnitt 2.3 beschrieben, ist es mit klassischen LBAC Richtlinien nicht möglich bestimmte Anforderungen an den Informationsfluss zu erfüllen. Dieses Problem kann mit DTE gelöst werden und DTE wiederum kann in g-SIS^c modelliert werden. Um eine vorhandene DTE Konfiguration in g-SIS^c zu überführen werden zunächst Gruppen für alle möglichen Typen erstellt, in denen sich die Objekte befinden. Für jede Domäne werden dann jeweils eine Gruppe für Benutzer und eine für Subjekte dieser Domäne erstellt. Die Benutzer besitzen die Mitgliedschaft in der Gruppe, die ihrer Domäne entspricht.

Für jede Domänengruppe mit Benutzern muss eine subordC Relation zur zugehörigen Gruppe der Subjekte dieser Domäne definiert werden. Damit ist es für Benutzer aus einer Domäne möglich, Subjekte in der entsprechenden Domäne zu erstellen. Die Relationen zwischen den Gruppen der Subjekte und den Gruppen der Objekte werden dann entsprechend der DTE Matrix definiert.

4.3 Role-Based Access Control in g-SIS^c

Außer LBAC und DTE kann auch Core RBAC in g-SIS^c modelliert werden. Jedoch ist das Ziel von g-SIS^c das Teilen von Informationen mit einem Fokus auf die Lese- und Schreibrechte eines Objekts. Da RBAC jedoch auch abstraktere Zugriffsrechte ermöglicht, ist es nicht sinnvoll dies direkt in g-SIS^c zu modellieren. Daher sollen in dieser Betrachtung die Zugriffsrechte von Objekten auf Lese- und Schreibrechte beschränkt sein. Es sei ein RBAC Modell mit den Rollen Manager und Experte gegeben. Die Rolle Manager hat Read und Write Zugriffsrechte auf das Objekt Bericht 1 und die Rolle Experte hat Read und Write Zugriffsrechte auf das Objekt Bericht 2. Dieses Modell ist in Abbildung 3 veranschaulicht.

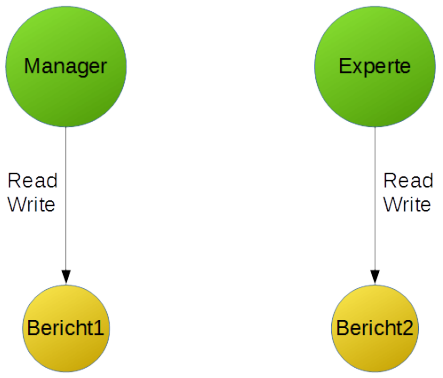

Figure 3: Ein Beispiel $RBAC_0$ Modell mit Lese- und Schreibrechten auf zwei Objekte

Soll dieses Modell nun in g-SIS konfiguriert werden, so muss zunächst für jede Rolle eine eigene Gruppe erstellt werden. Diese Gruppen repräsentieren die Zuordnung der Benutzer zu ihren Rollen. Jeder Benutzer darf jedoch nur in einer dieser Gruppen Mitglied sein. Da es in RBAC jedoch möglich ist, mehr als einer Rolle zuzugehören, müssen die Kombinationen der Rollen ebenfalls als Gruppen angelegt werden. In Abbildung 4 ist die resultierende g-SIS Konfiguration des $RBAC_0$ Modells dargestellt. Im Beispiel resultieren aus diesem Schritt die Gruppen Manager_G, Experte_G und als Kombination ManagerExperte_G.

Im nächsten Schritt müssen Gruppen für die möglichen Sessions angelegt werden. Dabei muss berücksichtigt werden, dass es möglich ist, dass eine Session keine aktivierten Rollen besitzt. Es muss also wiederum für jede möglche Kombination von aktivierten Rollen in einer Session eine Gruppe erstellt werden. Damit ergibt sich im Beispiel die Gruppe KeineRollen_S, für eine Session ohne aktive Rollen. Desweiteren Manager_S und Experte_S, für Sessions in denen die Manager oder die Experten Rolle aktiv ist, sowie die Gruppe ManagerExperte_S für den Fall, dass beide Rollen aktiviert sind.

Benutzer müssen in der Lage sein Subjekte in diesen Gruppen zu erstellen. Daher sind subordC Relationen zwischen den Rollengruppen und den Sessiongruppen definiert. Es muss außerdem in einer Session möglich sein Rollen zu aktivieren oder zu deaktivieren. Dazu müssen symmetrische subordM Relationen zwischen allen Sessiongruppen definiert werden.

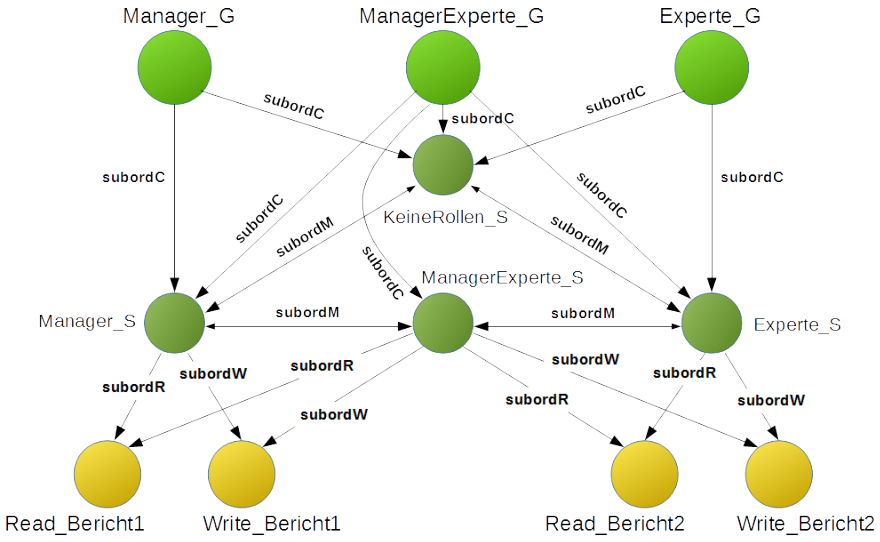

Figure 4: Die äquivalente g-SIS Konfiguration des $RBAC_0$ Modells

Das erlaubt den Subjekten die aktivierten Rollen zu ändern. Dies sollte jedoch nur möglich sein, wenn der Benutzer, der das Subjekt erstellt hat, auch den entsprechenden Rollen zugeordnet ist. Der Einfachheit halber soll hier jedoch auf diese Bedingungen nicht weiter eingegangen werden.

Als letzter Schritt müssen nun noch Gruppen für die Objekte gebildet werden. Jedes Objekt benötigt eine Gruppe für jedes Zugriffsrecht. Daher ergeben sich die Gruppen Read_Bericht1 und Write_Bericht1, sowie Read_Bericht2 und Write_Bericht2. Für jede Session werden dann entsprechend der aktiven Rollen die subordR und subordW Relationen zu den Objektgruppen definiert.

5 Fazit

References

[BSSW+95] L. Badger, D.F. Sterne, D.L. Sherman, K.M. Walker und S.A. Haghighat. Practical Domain and Type Enforcement for UNIX. In *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, May 1995, S. 66–77.

[FSGK+01] David F Ferraiolo, Ravi Sandhu, Serban Gavrila, D Richard Kuhn und Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 4(3), 2001, S. 224–274.

[KSNW09] Ram Krishnan, Ravi Sandhu, Jianwei Niu und William H. Winsborough. Foundations for Group-centric Secure Information Sharing Models. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, New York, NY, USA, 2009. ACM, S. 115–124.

[Sand93] R.S. Sandhu. Lattice-based access control models. *Computer* 26(11), Nov 1993, S. 9–19.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein und Charles E. Youman. Role-Based Access Control Models. *Computer* 29(2), Februar 1996, S. 38–47.

[SKNW10] Ravi Sandhu, Ram Krishnan, Jianwei Niu und WilliamH. Winsborough. Group-Centric Models for Secure and Agile Information Sharing. In Igor Kotenko und Victor Skormin (Hrsg.), *Computer Network Security*, Band 6258 der *Lecture Notes in Computer Science*, S. 55–69. Springer Berlin Heidelberg, 2010.