Analyse und Simulation dynamischer RBAC-Zugriffssteuerungsmodelle

Marius Schlegel

Technische Universität Ilmenau
marius.schlegel@tu-ilmenau.de

Zusammenfassung

Angriffe auf heutige IT-Systeme sind zunehmend komplexer und professioneller; oftmals begünstigt durch unzureichend oder sogar fehlerhaft realisierte Sicherheitseigenschaften. Ein Weg dieser Problematik zu begegnen und die korrekte Implementierung sicherheitskritischer Zugriffssteuerungssysteme zu erreichen, ist die Modellierung und Analyse auf Grundlage formaler Sicherheitsmodelle. Die Klasse der RBAC-Sicherheitsmodelle bietet eine adäquate Möglichkeit zur Modellierung weit verbreiteter und etablierter rollenbasierter Sicherheitspolitiken durch anwendungsnahe Abstraktionen und Paradigmen. Nachteilig ist jedoch, dass diese Modelle lediglich statische Rechtesituationen beschreiben und damit keine Analysen auf Sicherheitseigenschaften, die sich auf ihre dynamische Veränderlichkeit beziehen (z.B. die potentielle Ausbreitung von Zugriffsrechten), zulassen. Dieses Papier stellt einen Ansatz vor, klassische RBAC-Modelle um Komponenten des dynamischen Verhaltens zu erweitern und diese zum Gegenstand von Analysen auf dynamische Sicherheitseigenschaften zu machen. Darauf aufbauend wurde ein Werkzeug zur Simulation der Modelle für automatisierte Analysen entwickelt. Im Rahmen der Evaluierung wurde die praktische Machbarkeit des Ansatzes und die Praktikabilität des Werkzeugs bestätigt.

1 Einleitung

Heutige IT-Systeme sind einer nie dagewesenen Anzahl an Bedrohungen durch Schadprogramme, Sicherheitslücken, Identitätsdiebstähle und -missbräuche sowie zunehmend professionellen Angriffen ausgesetzt [BSI11, BSI14]. Diese kritische Bedrohungslage zwingt die Entwickler der Systeme einen besonderen Fokus auf die IT-Sicherheit zu legen. Für die Wahrung der Sicherheitseigenschaften Vertraulichkeit, Integrität und Verfügbarkeit spielen Zugriffssteuerungssysteme in Betriebssystemen, Datenbankmanagementsystemen und ERP-Systemen eine grundlegende Rolle. Unter einer Zugriffssteuerung versteht man die regelbasierte Kontrolle des Zugriffes von Subjekten (z.B. Nutzern, Prozessen) durch Operationen (z.B. lesen, schreiben) auf Objekte (z.B. Dateien, Sockets).

Diese Regelwerke werden durch sog. *Sicherheitspolitiken* beschrieben und durchgesetzt. Sicherheitspolitiken sind oftmals extrem kritische Systemkomponenten, sodass zu deren Entwicklung und zur Beförderung ihrer Qualität (z.B. Korrektheit und Konsistenz) formale Methoden eingesetzt werden. Um formale Analysen zu ermöglichen und folglich Sicherheitseigenschaften nachweisen oder widerlegen zu können, werden Sicherheitspolitiken in präzise, formale *Sicherheitsmodelle* [BeLa73, HaRU76, BrNa89, SCFY96, ShTr13] überführt.

In heutigen Zugriffssteuerungssystemen haben sich rollenbasierte Sicherheitspolitiken als zunehmend praxisrelevant erwiesen [EvBö04, SYRG07a, SYRG07b]. Nutzern sind dabei Rollen

P. Schartner · K. Lemke-Rust · M. Ullmann (Hrsg.) · D•A•CH Security 2015 · syssec (2015) 110-121.

(z.B. Arzt, Pfleger, Patient) zugeordnet, deren Innehaben über die Befugnis von Zugriffen auf Objekte entscheidet (z.B. Leserecht für elektronische Patientenakten). Um die formale Spezifikation solcher Politiken und folglich deren korrekte Implementierung zu ermöglichen, wurden die *Role-based-Access-Control-Modelle* (RBAC-Modelle) [SCFY96, Sand98], welche genau diese Abstraktionen bieten, geschaffen und standardisiert [ANS04]. Vorteilhaft ist hierbei, dass die Rollen der Nutzer direkt aus den Verantwortlichkeiten und Kompetenzen innerhalb einer Organisationsstruktur, z.B. der eines Unternehmens, abgeleitet werden können. Der entscheidende Nachteil der standardisierten RBAC-Modelle ist, dass diese jeweils nur ein Abbild des Zustandes eines Zugriffssteuerungssystems modellieren. Insbesondere ist es nicht möglich, Fragestellungen der Form „Erhält ein Sachbearbeiter (Nutzer) jemals Lesezugriff (Recht) auf ein nicht für ihn bestimmtes vertrauliches Dokument (Objekt) ausgehend von einer gegebenen Rechtesituation?“ zu beantworten. Für den Nachweis solcher Safety-Eigenschaften müssen RBAC-Modelle die dynamische Veränderung einer gegebenen Rechtesituation modellieren und damit analysierbar machen. Die vorliegende Arbeit widmet sich der Lösung dieser Problemstellung.

Ziel dieses Papiers ist es, die aufgezeigten Schwächen klassischer RBAC-Modelle zu beseitigen und diese für Analyseverfahren auf dynamische Sicherheitseigenschaften (insbesondere die potentielle Ausbreitung von Zugriffsrechten) zugänglich zu machen. Das vorliegende Papier leistet dazu folgende Beiträge: (1) die Ergänzung der RBAC-Modelle um Komponenten des dynamischen Verhaltens, (2) die Entwicklung einer Methode zur Analyse solcher Modelle, (3) die Implementierung eines Werkzeugs, das diese Methode realisiert, und (4) die Evaluierung des entwickelten Werkzeugs.

2 Methode

Das vorliegende Kapitel widmet sich der Methode zur Analyse rollenbasierter Zugriffssteuerungspolitiken auf potentielle Ausbreitungen von Zugriffsrechten mittels formaler Modelle. Dafür werden RBAC-Modelle als Grundlage, darauf aufbauend die entwickelten dynamischen RBAC-Modelle, sowie die Analysemethode selbst vorgestellt.

2.1 RBAC-Zugriffssteuerungsmodelle

Einen praxisbezogenen Ansatz der Modellierung heutiger Zugriffssteuerungspolitiken bietet die RBAC-Modellfamilie [SCFY96, Sand98, ANS04]. Jeder Nutzer erhält dabei eine Zuordnung zu einer Rolle oder mehreren Rollen. Diese Rollen können direkt aus den Kompetenzen innerhalb einer Organisationsstruktur abgeleitet werden. Beispielsweise existieren in einem Klinikmanagementsystem je Station die Rollen Oberarzt, Facharzt, Krankenpfleger und Patient. Durch ihre zusätzliche Assoziation zu Rechten entscheiden Rollen über die Erlaubnis, ob eine Zugriffsoperation auf ein bestimmtes Objekt durchgeführt werden darf oder nicht. Im genannten Beispiel besitzen Krankenpfleger der inneren Medizin (*NurseSurgery*) Lesezugriff (*view*) auf elektronische Patientenakten der auf der Station befindlichen Patienten (*RecentEPRSurgery*). Es müssen jedoch nicht immer alle Rollen, deren Mitglied ein Nutzer ist, auch aktiv sein. Diese Eigenschaft wird durch Sitzungen, welche Nutzern ihre gegenwärtig aktivierten Rollen zuordnen, repräsentiert (ähnlich einem Login). Ein Nutzer besitzt dann alle Rechte der in einer Sitzung aktiven Rollen. Außerdem kann ein Nutzer mehreren, verschiedenen Sitzungen angehören.

Das RBAC_0-Modell formalisiert genau diese grundlegenden Konzepte und wird durch ein Tupel $(U, R, P, S, UA, PA, user, roles)$ mit den wie folgt definierten Komponenten beschrieben.

- U ist die Menge aller Nutzer, R die Menge aller Rollen, $P = 2^{O \times OP}$ die Menge aller Rechte

mit O als Menge aller Objekte sowie OP als Menge aller Zugriffsoperationen, und S die Menge aller Sitzungen.

- $UA \subseteq U \times R$ ist die Nutzer-Rollen-Relation, welche die von Nutzern annehmbaren Rollen definiert. $PA \subseteq P \times R$ ist die Rechte-Rollen-Relation, welche die zu Rollen zugehörigen Rechte definiert.
- $user : S \to U$ ist die Abbildung jeder Sitzung $s_i \in S$ auf einen der Sitzung zugehörigen Nutzer $user(s_i)$. $roles : S \to 2^R$ ist die Abbildung jeder Sitzung $s_i \in S$ auf die während der jeweiligen Sitzung aktive Rollenmenge $roles(s_i) \subseteq \{r \in R \mid (user(s_i), r) \in UA\}$.

Die Zugriffsfunktion $f : U \times O \times OP \to \{true, false\}$ formalisiert nun die Bedingungen, unter welchen Zugriffe erlaubt oder verweigert werden; diese ist für $RBAC_0$-Modelle wie folgt definiert:

$$f(u,o,op) \mapsto \begin{cases} true, & \text{falls } \exists r \in R, s \in S : user(s) = u \wedge r \in roles(s) \wedge ((o,op),r) \in PA, \\ false, & \text{sonst.} \end{cases}$$

Im obigen Anwendungsszenario meldet sich eine Krankenpflegerin der inneren Medizin im Klinikmanagementsystem an und möchte auf die elektronische Patientenakte eines auf ihrer Station befindlichen Patienten lesend zugreifen. Die im System integrierte Zugriffsfunktion prüft dabei, ob die Nutzerin eine Sitzung besitzt, in der sie eine Rolle aktiviert hat, welcher die genannte Berechtigung zugewiesen ist. Da dies der Fall ist, wird ihr der Zugriff gewährt.

Auf dem $RBAC_0$-Grundmodell aufbauend erweitern $RBAC_1$- bzw. $RBAC_2$-Modelle den Kalkül jeweils unabhängig um Rollenhierarchien bzw. Restriktionen über einzelne Modellkomponenten zur Erhöhung der Ausdrucksstärke und zur Erweiterung der Modellierungsmöglichkeiten. Deren Vereinigung nimmt das konsolidierende $RBAC_3$-Modell vor [SCFY96, Sand98, ANS04].

2.2 Dynamische RBAC-Zugriffssteuerungsmodelle

Klassische RBAC-Modelle spiegeln jederzeit einen einzigen, statischen Zustand wider. Für die Durchführung von Analysen auf Sicherheitseigenschaften, die mit dem dynamischen Modellverhalten in Zusammenhang stehen (z.B. die potentielle Ausbreitung von Zugriffsrechten), ist dies jedoch nicht ausreichend. Um solche Analysen zu ermöglichen, verwenden klassische Modelle, z.B. HRU-Modelle [HaRU76], einen deterministischen Zustandsautomaten und auf diesem definierte Operationen, welche ausgehend vom gegenwärtigen Zustand durch Eingaben mögliche Übergänge in Folgezustände beschreiben. Durch Hinzufügen dieser Komponente können *dynamische RBAC-Modelle* (DRBAC-Modelle) auf dynamische Sicherheitseigenschaften analysiert werden. Im Folgenden wird ein Auszug der DRBAC-Modellfamilie vorgestellt, zu dem der Kalkül für DRBAC-Modelle, analog jenem der klassischen RBAC-Modelle, in der Bachelorarbeit [Schl13] entwickelt wurde.

Das zum $RBAC_0$-Modell korrespondierende $DRBAC_0$-Modell ist ein deterministischer Zustandsautomat (Q, Σ, δ, q_0) mit den folgenden Komponenten. Ein einzelner Zustand $q = (U_q, R_q, P_q, S_q, O_q, UA_q, PA_q, user_q, roles_q) \in Q$ stellt ein Abbild der RBAC-Komponenten dar. Die Zustandsmenge Q enthält alle dynamisch veränderlichen Komponenten des klassischen $RBAC_0$-Modells. Der initiale Zustand $q_0 \in Q$ bildet den Startzustand des Automaten. Die gewünschte Dynamik des Modells wird durch die Zustandsübergangsfunktion $\delta : Q \times \Sigma \to Q$ beschrieben (oftmals auch als Autorisierungsschema bezeichnet). Diese wird durch eine Reihe von Ausdrücken definiert, sog. Kommandos, welche genau die Möglichkeiten des modellierten

Zugriffssteuerungssystems und damit mögliche Zustandsveränderungen unter Angabe des Ausgangszustandes und den Parametern der Eingabemenge Σ beschreiben.

Vor Ausführung eines anwendungsspezifischen Kommandos der betrachteten DRBAC-Modellinstanz ist es notwendig, eine Reihe von Bedingungen (*Bedingungsteil*) zu prüfen, um anschließend den Folgezustand durch eine Folge von sog. Primitivoperationen (*Anweisungsteil*), die die Änderung der einzelnen Modellkomponenten definieren, zu beschreiben.

Obligatorisch ist die von den RBAC-Modellen bekannte Abfrage der Zugriffsfunktion für jegliche Zugriffe. Zusätzlich dazu enthält der Bedingungsteil eines Kommandos eine Menge an wahlfreien Bedingungen, welche die Existenz eines Elements oder mehrerer Elemente in den Modellkomponenten des eingegebenen Zustandes prüfen. Wahlfreie Bedingungen können zum einen konjunktiv mit dem obligatorischen Bedingungsteil sowie zum anderen konjunktiv untereinander verknüpft auftreten.

Aus Platzgründen beschränkt sich dieser Abschnitt exemplarisch auf die Definition der obligatorischen Bedingung $cond_{RBAC}$ (kann zusätzlich auch wahlfrei auftreten) und der wahlfreien Bedingung $cond_{UA}$ zur Prüfung, ob einem Nutzer u eine bestimmte Rolle r zugewiesen ist.

- $\text{cond}_{\text{RBAC}} : Q \times U \times O \times OP \to \{true, false\}$ mit
$$\text{cond}_{\text{RBAC}}(q,u,o,op) \mapsto \begin{cases} true, & \text{falls } \exists r \in R_q, s \in S_q : user_q(s) = u \ \wedge \ r \in roles_q(s) \ \wedge \\ & \quad ((o,op),r) \in PA_q, \\ false, & \text{sonst.} \end{cases}$$
- $\text{cond}_{\text{UA}} : Q \times U \times R \to \{true, false\}$ mit
$$\text{cond}_{\text{UA}}(q,u,r) \mapsto \begin{cases} true, & \text{falls } (u,r) \in UA_q, \\ false, & \text{sonst.} \end{cases}$$

Beispielsweise prüft ein anwendungsspezifisches Kommando *assignMedicalTeamRoleToNurse* für die Zuweisung einer Pflegekraft zu einem medizinischen Team neben den notwendigen Berechtigungen des Aufrufers (dieser muss die Rolle *MedicalManager* aktiviert haben), ob auch die Pflegekraft die geforderte Rolle *Nurse* annehmen darf (laut Modellkomponente *UA*).

Erst wenn alle Bedingungen des gesamten Bedingungsteils als zutreffend ausgewertet wurden, können die Primitivoperationen im Anweisungsteil ausgeführt werden, um zu einem Zustandswechsel zu führen. Für diese Primitiven lässt sich eine Unterteilung in nutzerbezogene, rollenbezogene, rechtebezogene, sitzungsbezogene und objektbezogene Primitivoperationen vornehmen. Im Wesentlichen werden damit Hilfsoperationen zur Verfügung gestellt, die auf die korrekte Manipulation der Modellkomponenten, wie das Hinzufügen und Entfernen von Elementen über diesen Komponenten und deren Abhängigkeiten, abzielen.

Exemplarisch erfolgt die Definition der Primitivoperation *assignRoleToUser*, welche einem Nutzer u eine Rolle r zuweist, falls er dieser Rolle noch nicht angehört.

- $\text{assignRoleToUser} : Q \times U \times R \to Q$ mit
$\text{assignRoleToUser}(q,u,r) \mapsto (U_q, R_q, P_q, S_q, O_q, UA_q \cup \{(u,r)\}, PA_q, user_q, roles_q)$,
wobei $u \in U_q, r \in R_q, (u,r) \notin UA_q$.

Beispielsweise veranlasst das Kommando *assignMedicalTeamRoleToNurse* mittels der Primitive *assignRoleToUser* die Zuweisung der Rolle *MedicalTeam* und, falls diese im Ausgangszu-

stand noch nicht zugewiesen war, die Veränderung der Komponente *UA* sowie den Übergang in einen neuen Folgezustand.

Den obigen Erläuterungen folgend wird das Kommando *assignMedicalTeamRoleToNurse* wie folgt definiert.

- $\delta(q, \text{assignMedicalTeamRoleToNurse}, u, \mathit{MedicalManager}, x_u) ::=$
  if $\text{cond}_{\text{RBAC}}(q, u, \mathit{MedicalManager}, \mathit{assignMedicalTeamRoleToNurse}) \wedge$
  $\text{cond}_{\text{UA}}(q, x_u, \mathit{Nurse})$
  then $\text{assignRoleToUser}(q, x_u, \mathit{MedicalTeam})$
  end if

Korrespondierend zu den erweiterten RBAC-Modellen wurden ebenfalls Erweiterungen für den vorgestellten DRBAC-Kalkül entwickelt, welche weitere Modellkomponenten und zusätzliche Primitivoperationen für Rollenhierarchien und Restriktionen bieten [Schl13].

2.3 Analyse auf Safety-Eigenschaften

Die im vorherigen Abschnitt präsentierte Modellierung von rollenbasierten Zugriffssteuerungspolitiken ermöglicht nun Analysen auf dynamische Sicherheitseigenschaften. Im Vergleich zu HRU-Modellen müssen potentielle Ausbreitungen von Zugriffsrechten für DRBAC-Modelle weitreichender betrachtet werden, weil diese Modelle zusätzliche Abstraktionen und Ausdrucksmittel kennen, welche wiederum mit vielfältigeren Wegen von Rechteausbreitungen einhergehen.

Prinzipiell können Berechtigungen durch neue Zuordnungen von Rechten, Rollen oder Sitzungen ausgehend von einem bestimmten Zustand des untersuchten Modells ausgebreitet werden. Als Ziel der Analyse eines DRBAC-Modells lassen sich also spezielle Elemente der auf Rechte, Rollen und Sitzungen bezogenen Modellkomponenten betrachten. Korrespondierend zu den vorgestellten Primitivoperationen können Safety-Eigenschaften der *Permission-Safety*, *Role-Safety*, *Session-Safety* und der vereinenden *DRBAC-Model-Safety* definiert werden [Schl13].

Für dieses Papier wird, aufgrund der intuitiven Analogie zur rechtebezogenen HRU-Safety, exemplarisch die folgende Definition der Permission-Safety betrachtet.

> Ein Zustand q eines gegebenen DRBAC-Modells ist *permission-safe* in Bezug auf ein Recht $p \in P_q$ genau dann, wenn beginnend mit q keine Folge von Eingaben existiert, die dieses Recht p zusammen mit einer Rolle $r \in R_q$ in *PA* einträgt, welche dieses Tupel nicht schon in q enthält.

Es lässt sich erkennen, dass einzig die Primitivoperation *assignPermissionToRole*, die der Zuweisung von Rechten zu Rollen dient, zur Verletzung dieser Eigenschaft führen kann. Beispielsweise könnte im bereits genannten Klinikmanagementsystem ein Nutzer in der Rolle des Doktors (*DoctorSurgery*) bei eigener Abwesenheit allen Krankenpflegerinnen und -pflegern (*Nurse-Surgery*) auf seiner Station (temporär) die Berechtigung für die Aktualisierung der Krankenakten anwesender Patienten (*update*) erteilen. Möglicherweise ist jedoch eine solche Ausbreitung von Zugriffsrechten innerhalb des Modells unerwünscht und würde somit die Sicherheitseigenschaft der Permission-Safety bezüglich der Aktualisierungsberechtigung verletzen.

Wie die HRU-Safety-Eigenschaft sind auch die Safety-Eigenschaften der DRBAC-Modelle für einen jeweils konkreten Zustand nicht entscheidbar. Folglich existiert ebenso kein allgemein gültiger Algorithmus zur Lösung dieses Problems. Ein möglicher und in dieser Arbeit verfolgter

Ansatz ist die Simulation von DRBAC-Modellen und die Suche nach einem Zustand, welcher die untersuchte Safety-Eigenschaft verletzt. Bei Fund eines solchen unsicheren Zustandes lässt sich auf diese Weise die Safety des analysierten Modells widerlegen.

Insbesondere für komplexe realitätsnahe Modelle ist die reine *Brute-Force*-Suche nach unsicheren Zuständen wenig erfolgversprechend. Idee ist es hier, unter Anwendung von heuristischen Algorithmen, ähnlich denen, die bereits für HRU-Modelle existieren, einen Pfad vom Ausgangszustand q zu einem Zustand q', in dem die untersuchte Safety-Eigenschaft nicht gilt, zu konstruieren und damit zu beweisen, dass der Zustand q' Teil des Zustandsraums des gegebenen Modells ist und folglich die Safety-Eigenschaft des Modells nicht gilt.

Ein anwendbarer Algorithmus ist die *Dependency-Search*-Heuristik [AmKP13]. Im Wesentlichen werden durch eine statische Voranalyse des Autorisierungsschemas in ihrer Ausführung voneinander abhängige Kommandos identifiziert und darauf aufbauend ein Pfad zum unsicheren Zustand durch schrittweise Erfüllung der Bedingungen konstruiert.

Die Methode führt dann in jedem Schritt unter Eingabe aller notwendigen Parameter (Ausgangszustand, Kommandobezeichner, Parameter des Kommandos) durch die Heuristik das entsprechende anwendungsspezifische Kommando (als Teil des Authorisierungsschemas des Modells) aus. Für die Einhaltung der untersuchten Safety-Eigenschaft wird bei jedem so erzeugten neuen Zustand geprüft, ob während des Zustandswechsels die entsprechende Primitive, die die Verletzung dieser Safety-Eigenschaft verursacht, ausgeführt wurde.

3 Werkzeug

Die Analyse rollenbasierter Zugriffssteuerungspolitiken nach der vorgestellten Methode ist je nach modelliertem System sehr komplex; die manuelle Durchführung von Analysen ist daher ausgeschlossen. Aus diesem Grund wurde ein entsprechendes Werkzeug zur Automatisierung der Methode für die Analyse von dynamischen RBAC-Zugriffssteuerungsmodellen auf Safety-Eigenschaften entwickelt.

Funktionale Beschreibung

Das Werkzeug integriert sich nahtlos in WorSE (*Workbench for Model-based Security Engineering*) [AmKP14] – eine Plattform zur modellbasierten Entwicklung, Spezifikation und Analyse von Sicherheitspolitiken. In dieser Plattform sind Funktionen zur Spezifikation und Integration von Modellen, zur Analyse (Simulatoren, Analyseheuristiken) und zur grafischen Darstellung (GUI, Visualisierung der Analyseergebnisse, Assistenten zum Projektmanagement) vereint. Das entwickelte Werkzeug erweitert somit die Funktionsgruppen der Modellintegration sowie der Analysewerkzeuge.

Softwarearchitektur

WorSE bietet eine modulare, pluginbasierte Architektur. Es besteht die Möglichkeit, Komponenten zu Modellentwurf, Modellspezifikation und Modellanalyse separat zu implementieren und einzubinden. Die einzelnen Komponenten interagieren über ein offenes, blackboardartiges Kommunikationsmodell (*Tuple Space* [Gele89]) miteinander. Auf Basis dieser Schnittstellen wurde das Werkzeug zur Simulation von DRBAC-Modellen geschaffen. Abbildung 1 stellt die Architektur von WorSE schematisch dar.

Das Fundament der Realisierung bildet die Softwarekomponente *Modell*, welche die Abstrak-

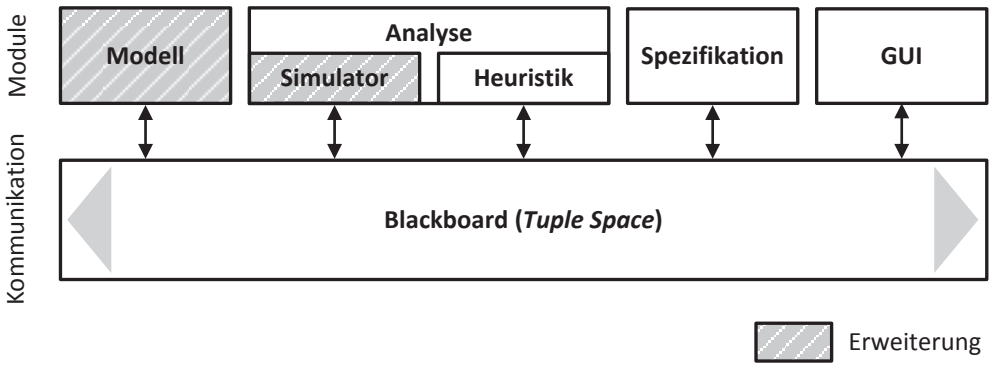

Abb. 1: Schematische Darstellung der WorSE-Architektur

tionen des DRBAC-Modellkalküls (Modellkomponenten und Autorisierungsschema) in implementierungstechnische Entsprechungen überführt. Hierbei wurde besonders auf die Effizienz der verwendeten Datenstrukturen Wert gelegt, um geringe Zugriffszeiten auf die internen Darstellungen und damit gute Laufzeiten des Werkzeugs zu ermöglichen. Der *Simulator* als Softwarekomponente zur Simulation des Autorisierungsschemas nutzt diese Datenstrukturen und prüft in jedem Simulationsschritt für die Ausführung eines anwendungsspezifischen Kommandos zunächst den Bedingungsteil. Sind alle Bedingungen erfüllt, so wird die Ausführung aller im Anweisungsteil aufgeführten Primitivoperationen getätigt. Wie in Abschnitt 2.3 beschrieben, verursacht die Ausführung bestimmter Primitiven zusammen mit den zugehörigen Parametern die Verletzung bestimmter Safety-Eigenschaften. Die Prüfung dieses Sachverhaltes wird ebenfalls vom Simulator vorgenommen.

Im Rahmen einer Analyse kann ein konkretes DRBAC-Modell über die grafische Oberfläche (GUI) von WorSE geladen werden. Das Modell, der Initialzustand und die Analyseparameter (z.B. die zu untersuchende Safety-Eigenschaft und die zu verwendende Heuristik) liegen dabei in einer Datei im XML-Format vor und werden zusammen mit der dem Werkzeug beigelegten XML-Schema-Definition für DRBAC-Modelle eingelesen. Für die Safety-Analyse selbst arbeiten Heuristik und Simulator im ständigen Wechsel. In jedem Simulationsschritt generiert die verwendete Heuristik die Eingabe für den Simulator bestehend aus einem Ausgangszustand q, einem anwendungsspezifischen Kommando op, einem Nutzer u (welcher die Zugriffsanfrage stellt), einem Objekt o (als Zugriffsziel) und einem zugehörigen Parametervektor x. Der Simulator nimmt dann die Ausführung vor, erzeugt einen Folgezustand, prüft die Verletzung der untersuchten Safety-Eigenschaft und übergibt der Heuristik das Ergebnis. Falls eine Rechteausbreitung aufgetreten ist, wird die Simulation beendet, da die Nichteinhaltung der Safety-Eigenschaft im untersuchten DRBAC-Modell nachgewiesen werden konnte. Ansonsten wiederholt sich dieser Ablauf im nächsten Simulationsschritt erneut, solange bis eine Rechteausbreitung auftritt oder eine initial definierte Höchstschrittzahl erreicht wird. Das Ergebnis der Safety-Analyse kann in Textform (Protokoll) oder grafischer Form (Zustandsbaum) ausgelesen werden und so als Grundlage der Verbesserung des DRBAC-Modells und damit der Sicherheitspolitik dienen.

Das Werkzeug wurde in C++ unter Verwendung frei verfügbarer Bibliotheken für die Bereitstellung der Möglichkeiten zur Nutzerinteraktion (*Qt*, *log4cxx*), zum Parsen von XML-Dateien (*Xerces-C++*) und zur Manipulation der Zustandsgraphen (*Boost*) implementiert.

4 Evaluierung

Das vorliegende Kapitel widmet sich nun der Evaluierung der erreichten Ergebnisse. Ziel ist es, die praktische Machbarkeit der entwickelten Methode zur simulativen Analyse von DRBAC-Zugriffssteuerungsmodellen nachzuweisen. Der Gegenstand dieser Evaluierung ist das implementierte Werkzeug, welches hinsichtlich seiner Laufzeitperformanz untersucht wird.

4.1 Evaluierungsmethode

Zur Evaluierung wurden Laufzeitmessungen durchgeführt. Um die daraus ermittelten Ergebnisse angemessen darstellen und bewerten zu können, bedarf es Kriterien der Evaluierung. Die zwei folgenden Maßzahlen werden als solche verwendet: die *effektive Schrittzeit* (ESZ) und der *Durchsatz*. Die ESZ gibt die mittlere Schrittzeit für einen effektiven, zustandsverändernden Simulationsschritt an. Unter dem Durchsatz ist in dieser Arbeit die mittlere Anzahl der Zustandsveränderungen pro Zeiteinheit zu verstehen. Im Rahmen der Messungen wird nur die ESZ ermittelt, denn der Durchsatz lässt sich aus dem Reziproken der ESZ berechnen.

Die praktische Machbarkeit wurde anhand des DRBAC-Modells eines Gesundheitsinformationssystems evaluiert. Das Modell basiert auf den Untersuchungen einer Fallstudie im Kontext einer Betreuungseinrichtung für Senioren [EvBö04] und einer darauf aufbauenden, rollenbasierten Sicherheitspolitik [SYRG07a, SYRG07b]. Diese Politik wurde als Fallbeispiel in der Dissertation [Pölc14] aufgegriffen und für ein automatenbasiertes Modell konkretisiert. Nach Anpassung des Modells an den Kalkül und die Notationen der vorliegenden Arbeit besitzt das resultierende DRBAC_3-Modell 10 Rollen, 12 Objekte und 15 Kommandos innerhalb des Autorisierungsschemas [Schl13]. Gemessen wurde die Laufzeit jedes Kommandos gemittelt über fünf Durchläufe. Um zusätzlich die Laufzeiten für die Prüfung einzelner Bedingungen und die Ausführung einzelner Primitiven zu ermitteln und Aussagen über diese treffen zu können, erfolgte zusätzlich die Erstellung eines künstlichen DRBAC-Modells, in dessen Autorisierungsschema jedes Kommando jeweils eine Art der Bedingungen oder Primitiven enthält.

Die ESZ wurde mithilfe von berechneten Zeitdifferenzen ermittelt. Einen für diesen Zweck nützlichen Systemaufruf stellt der Linux-Kernel selbst bereit: *clock_gettime()* [Man10]. Die Funktion unterstützt eine Auflösung im Nanosekundenbereich wie auch verschiedene Uhren, die zur Messung angewendet werden können. In dieser Evaluierung wurde *CLOCK_PROCESS_CPUTIME_ID* zur Messung der virtuellen Prozesszeit verwendet. Um möglicherweise auftretende Caching-Effekte („Cache Misses“) zu minimieren, sind in jedem Messdurchlauf 20.000 Iterationen durchgeführt worden, über deren benötigte Ausführungszeit anschließend gemittelt wurde.

Alle Simulationsläufe wurden auf einem zeitgemäßen Desktop-Computersystem (*Intel® Core™ i7-2600* 3,4 GHz CPU, 16 GiB 1.333 MHz DDR3-RAM) unter Ubuntu 12.04 LTS ausgeführt.

4.2 Evaluierungsergebnisse

Die Ergebnisse der Messungen des ersten Testfalls sind Abbildung 2 zu entnehmen.

In der Grafik lassen sich unter den Kommandos zunächst einige „Ausreißer“ (ESZ > 30 µs) erkennen, deren ESZ oberhalb des Mittels der anderen Kommandos von aufgerundet 17 µs liegt. Dies hat seine Ursachen in der Verwaltung der Modellmengen und wird bei der Diskussion der Ergebnisse des zweiten Modells näher begründet. Die insgesamt resultierende Laufzeitperformanz stützt die praktische Eignung der DRBAC-Modelle für simulative Analyseverfahren: der

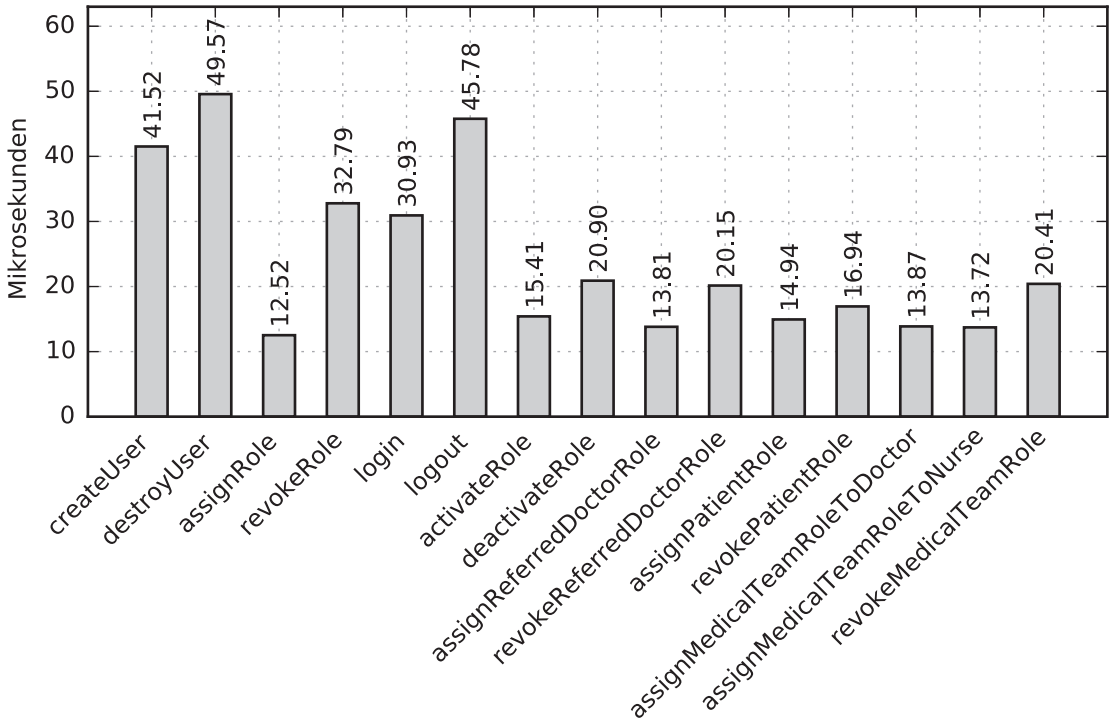

Abb. 2: Effektive Schrittzeiten der Kommandos des Modells eines Gesundheitsinformationssystems

DRBAC-Modellsimulator erreicht über alle Kommandos des vorliegenden Autorisierungsschemas gemittelt einen Durchsatz von ca. 42.000 Ausführungen pro Sekunde. Für die Ausführung eines einzelnen Kommandos, welches in einem effektiven Schritt resultiert, benötigt der Simulator im Mittel rund 24 µs (ESZ). Mit einer entsprechend effizienten Heuristik lassen sich somit angemessene Laufzeiten für Analysen realistischer DRBAC-Modelle erreichen.

Von Interesse ist außerdem, welchen Einfluss einzelne Bedingungen und Primitiven auf die Laufzeit eines gesamten Kommandos haben. Die Ergebnisse sind in Abbildung 3 dargestellt.

Das Werkzeug schafft es im Mittel ca. 80.000 Primitivoperationen pro Sekunde auszuführen und ca. 150.000 Bedingungen pro Sekunde zu prüfen. Daraus ergibt sich eine ESZ von rund 12,5 µs pro Ausführung bzw. rund 6,7 µs pro Prüfung. Es lässt sich erkennen, dass die Ausführung von *create*- bzw. *destroy*-Primitiven besonders hohe Laufzeiten verursacht. Dieses Phänomen ist zum einen damit begründbar, dass Nutzer, Rollen und Sitzungen neben den eigentlichen physischen Objekten ebenfalls als Objekte und damit als potentielle Ziele von Zugriffsoperationen im formalen Modell angesehen werden. Daher folgt innerhalb der Implementierung des Werkzeugs dem Einfügen des entsprechenden Elements eine weitere Einfügeoperation in die Menge aller Objekte, welche eine logarithmische Komplexität $\mathcal{O}(\log_2 n)$ aufweist. Zum anderen muss beim Erzeugen eines solchen Objektes die implementierte Menge aller Berechtigungen konsistent gehalten werden, worauf nochmals eine Einfügeoperation mit logarithmischer Laufzeit in Komplexität $\mathcal{O}(\log_2 n)$ verwendet wird. Für die *destroy*-Primitiven gilt dies analog.

Jedoch werden *create*- wie auch *destroy*-Primitiven im Rahmen von Safety-Analysen von intelligenten Heuristiken nur im beschränkten Maße ausgeführt, da diese nicht direkt, sondern lediglich in Kombination mit *assign*- oder *activate*-Primitiven innerhalb von Kommandos für

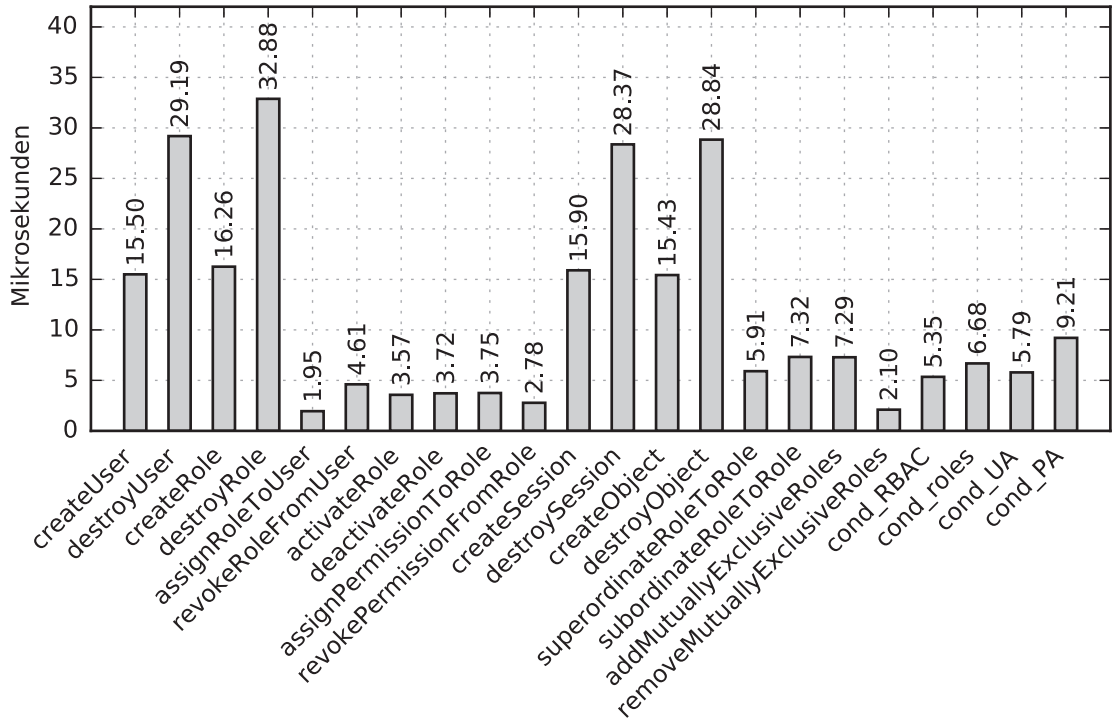

Abb. 3: Anteilige effektive Schrittzeiten der Bedingungen und Primitiven des DRBAC-Modellkalküls

potentielle Rechteausbreitungen in Frage kommen. Letztere zwei Arten von Primitiven benötigen vielfach geringere Laufzeiten. Daher sind auch die Spitzenlaufzeiten der Kommandos des ersten Modells weniger stark zu gewichten, weil diese in der Ausführung der Safety-Analyse letztlich auch nur einen relativ geringen Einfluss auf die Laufzeit haben. Dies bestätigt die These, dass praktische und realitätsbezogene DRBAC-Modelle mit dem entwickelten Werkzeug effizient simulierbar sind.

In den vorgestellten Messergebnissen wurde ein zusätzlicher Einfluss auf die Laufzeit des Simulators noch nicht betrachtet: die Duplikatsprüfung. Diese kann je nach Anzahl bisher erreichter Zustände – was bereits durchgeführte Tests bestätigen – einen erheblichen Teil der Laufzeit eines Analyse- bzw. Simulationsvorganges ausmachen, denn jeder bisher erreichte Zustand muss mit dem aktuell neuen Zustand verglichen werden. Für die Prüfung eines Zustandes, ob dieser ein Duplikat eines schon erreichten Zustandes ist, lässt sich $\mathcal{O}(|\textit{Anzahl bisher erreichter Zustände}|)$ als asymptotisch obere Laufzeitschranke angeben. Ebenfalls einen Einfluss auf die Laufzeit des Werkzeugs und die Simulation eines DRBAC-Modells im Speziellen haben: Umfang und Komplexität des Modells, Umfang und Komplexität des Autorisierungsschemas, sowie die Anzahl erreichbarer Zustände.

5 Zusammenfassung

Dieses Papier widmete sich dem Problem, dass RBAC-Modelle heute vorherrschender rollenbasierter Sicherheitspolitiken lediglich statische Rechtesituationen beschreiben, jedoch sich deren dynamische Veränderung nicht modellieren lässt und daher Analysen dynamischer Sicherheitseigenschaften mittels formaler Methoden nicht möglich sind.

Für die Lösung des Problems wurden dynamische RBAC-Zugriffssteuerungsmodelle durch die Ergänzung der klassischen RBAC-Modelle um einen Zustandsautomaten geschaffen, deren Safety-Eigenschaften formuliert und die Methode zur Analyse von DRBAC-Modellen vorgestellt. Zur Automatisierung der Methode für Analysen dynamischer RBAC-Modelle auf Safety-Eigenschaften wurde ein entsprechendes Werkzeug entwickelt. Das Werkzeug integriert sich in WorSE – einem Software-Framework zur modellbasierten Entwicklung, Spezifikation und Analyse von Sicherheitspolitiken. Die Ergebnisse der Evaluierung bestätigen, dass realitätsbezogene DRBAC-Modelle mit dem entwickelten Werkzeug effizient simulierbar sind, und zeigen damit die praktische Machbarkeit der Methode.

Um den auch in Zukunft immer komplexeren Systemen gerecht zu werden, kann zum einen die Mächtigkeit der Ausdrucksmittel des Modells erhöht werden, beispielsweise um weitere Restriktionen [CXOL+07] oder zusätzliche Möglichkeiten von Rollenhierarchien [Cram03], welche jeweils in erweiterten DRBAC-Modellen von Bedeutung sind. Zum anderen kann das entwickelte Werkzeug aufgrund der hohen Parallelisierbarkeit der Methode zur Steigerung der Laufzeitperformanz um eine Unterstützung von Mehrkern-Prozessorarchitekturen für parallele Safety-Analysen ergänzt werden.

Literatur

[AmKP13] P. Amthor, W. E. Kühnhauser, A. Pölck: Heuristic Safety Analysis of Access Control Models. *In: Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, SACMAT '13, New York, NY, USA (2013), 137–148.

[AmKP14] P. Amthor, W. E. Kühnhauser, A. Pölck: WorSE: A Workbench for Model-based Security Engineering. In: *Computers & Security*, 42, 0 (2014), 40–55.

[ANS04] American National Standards Institute (ANSI): Role Based Access Control (2004), ANSI/INCITS 359-2004.

[BeLa73] D. E. Bell, L. J. LaPadula: Secure Computer Systems: Mathematical Foundations. Technischer Bericht MTR-2547, Volume I, MITRE Corporation, Bedford, MA (1973).

[BrNa89] D. F. C. Brewer, M. J. Nash: The Chinese Wall Security Policy. *In: Proceedings of the 1989 IEEE Symposium on Security and Privacy*, IEEE Press (1989), 206–214.

[BSI11] Die Lage der IT-Sicherheit in Deutschland 2011. Lagebericht, Bundesamt für Sicherheit in der Informationstechnik (2011), URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2011_nbf.pdf.

[BSI14] Die Lage der IT-Sicherheit in Deutschland 2014. Lagebericht, Bundesamt für Sicherheit in der Informationstechnik (BSI) (2014), URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf.

[Cram03] J. Crampton: On Permissions, Inheritance and Role Hierarchies. *In: Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, ACM, New York, NY, USA (2003), 85–92.

[CXOL+07] D. W. Chadwick, W. Xu, S. Otenko, R. Laborde, B. Nasser: Multi-session Separation of Duties (MSoD) for RBAC. *In: Proceedings of the 2007 IEEE 23rd Interna-*

tional Conference on Data Engineering Workshop, ICDEW '07, IEEE Computer Society (2007), 744–753.

[EvBö04] M. Evered, S. Bögeholz: A Case Study in Access Control Requirements for a Health Information System. *In: Proceedings of the Second Workshop on Australasian Information Security, Conferences in Research and Practice in Information Technology*, Australian Computer Society, Inc. (2004), *ACSW Frontiers '04*, Bd. 32, 53–61.

[Gele89] D. Gelernter: Multiple Tuple Spaces in Linda. *In: Proceedings of the Parallel Architectures and Languages Europe, Volume II: Parallel Languages*, PARLE '89, Springer-Verlag, London, UK (1989), 20–27.

[HaRU76] M. A. Harrison, W. L. Ruzzo, J. D. Ullman: Protection in Operating Systems. In: *Communications of the ACM*, 19, 8 (1976), 461–471.

[Man10] clock_gettime(2) Manpage (Ubuntu 12.04 LTS). URL: http://manpages.ubuntu.com/manpages/precise/en/man2/clock_gettime.2.html (2010).

[Pölc14] A. Pölck: Small TCBs of Policy-controlled Operating Systems. Universitätsverlag Ilmenau (2014).

[Sand98] R. S. Sandhu: Role-Based Access Control. In: *Advances in Computers*, 46 (1998), 237–286.

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman: Role-Based Access Control Models. In: *IEEE Computer*, 29, 2 (1996), 38–47.

[Schl13] M. Schlegel: Symbolische Ausführung dynamischer RBAC-Zugriffssteuerungsmodelle. Bachelorarbeit, Technische Universität Ilmenau (2013).

[ShTr13] A. Sharifi, M. V. Tripunitara: Least-Restrictive Enforcement of the Chinese Wall Security Policy. *In: Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, SACMAT '13, ACM, New York, NY, USA (2013), 61–72.

[SYRG07a] S. D. Stoller, P. Yang, C. R. Ramakrishnan, M. I. Gofman: Efficient Policy Analysis for Administrative Role Based Access Control. *In: Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, ACM, New York, NY, USA (2007), 445–455.

[SYRG07b] S. D. Stoller, P. Yang, C. R. Ramakrishnan, M. I. Gofman: RBAC and ARBAC Policies for a Small Health Care Facility. URL: http://www.cs.sunysb.edu/~stoller/ccs2007/healthcare.txt (2007).