Universität Potsdam
Institut für Informatik
Sommersemester 2004
Prof. Dr. K. Rebensburg
Dipl. Inf. T. J. Wilke

Seminar

Trust Management und Security Policy Enforcement

Autoren: Lisa Böhringer, Dennis Brockhoff, Manuel Dräger, Kolja Engelmann, René Freitag, Mathias Fritzsche, Ronny Gerlach, Tobias Grave, Jörn Hartwig, Christian Herrmann, Martin Hoffmann, Jurrack Steffen, Daniel Kaulbars, Ronny Müller, Alexander Renneberg, Wieland Rhenau, Eldar Sultanow, Tobias Tebner

Datum: Potsdam, 16. Juli 2004

Inhaltsverzeichnis

1 Konzepte und Modelle für Design, Formalisierung und Verifikation von Sicherheitspolitiken

Tobias Tebner, Ronny Gerlach

Abstract

Dieses Paper beschreibt das Zusammenspiel zwischen Sicherheitspolitiken, Sicherheitsmodellen und den zu Grunde liegenden Sicherheitskriterien. Dabei werden zunächst die Entwicklung, die bestehenden Arten, internationale Richtlinien sowie die Zielgruppen von Sicherheitskriterien erläutert. Im Anschluss werden Sicherheitsmodelle hinsichtlich ihrer Klassifikation und ihrer Arten untersucht. Ebenso wird auf die Verifikationsmöglichkeiten der Sicherheitsmodelle mit verschiedenen Verifikationsmethoden eingegangen. Mit der Betrachtung einer Beispiel Sicherheitsarchitektur in Form einer "Security Suite", hinsichtlich der Architektur und Implementation, schließt dieses Paper ab.

1.1 Einleitung

Betrachtet man heutige Netzwerkstrukturen, fällt als erstes die in den letzten Jahren enorm gewachsene Komplexität auf. Immer größere Netzwerke müssen administriert, gewartet und geschützt werden. Gerade aufgrund dieser gestiegenen Komplexität sind "stand-alone" Sicherheitslösungen wie Firewalls, Virenscanner o.ä. nicht mehr ausreichend. Es ist ein Zusammenspiel einer Mehrzahl von Sicherheitskomponenten nötig, um einen systemweiten Schutz zu garantieren. An dieser Stelle kommen die Sicherheitspolitiken ins Spiel. Damit kann der Administrator eines Netzwerkes genau festlegen, zu welchen Aktionen die einzelnen Subjekte und Objekte eines Netzwerks berechtigt sind und zu welchen nicht. Allerdings nützt eine Sicherheitspolitik nicht viel, wenn sie nicht konsequent und systemweit durchgesetzt wird. Hier setzen die Sicherheitsmodelle an, welche durch eine zentrale Verwaltung und einer Kombination unterschiedlichster Sicherheitskomponenten für die Durchsetzung der Sicherheitspolitiken sorgen. Dazu zählen u.a. intelligente Reporting-Funktionen, welche durch eine zentrale Event-Management-Komponente ausgewertet werden, woraufhin dann anhand von administrativen Vorgaben entschieden wird, wie auf bestimme Bedrohungen reagiert werden soll. Im Laufe der Ausarbeitung wird schließlich aufgezeigt, wie sich das Zusammenspiel der zuvor genannten Komponenten im Einzelnen gestaltet, beginnend mit einer Betrachtung bestehender internationaler Sicherheitskriterien.

1.2 Sicherheitskriterien und ihre Bedeutung

Sicherheitskriterien stellen Richtlinien für die Entwicklung von sichereren und vertrauenswürdigen Systemen dar und bilden zudem die Grundlage für Sicherheitspolitiken. Sie sollen eine objektive Bewertung dieser Systeme von einer neutralen Instanz aus ermöglichen. Ebenso soll für Anwender und Benutzer die Möglichkeit geschaffen werden, IT-Sicherheitsprodukte gemäß den vorhandenen Sicherheitsanforderungen auszuwählen.

1.2.1 geschichtliche Entwicklung

Im Jahr 1983 gab das amerikanische Verteidigungsministerium die erste Version ihres „Orange Books“ (TCSEC) heraus. Dies war der Vorreiter für alle weiteren Arbeiten an solchen Kriterien. 1989 starteten Deutschland, Frankreich, Großbritannien und Niederlande unter deutscher Leitung ihre Arbeit an gemeinsamen europäischen Kriterien mit den Namen ITSEC. Die Version 1.0 wurde 1990 fertig gestellt. 1993 konstituierte sich dann das Common Criteria Editorial Board CCEB mit Vertretern aus aller Welt und ihrem gleichnamigen Sicherheitsstandard. 1996 beschlossen die ISO - Arbeitsgruppen JTC 1, SC 27 und WG 3, ihre bisherigen Normentwürfe zusammenzuführen und durch die entsprechenden Teile der Common Criteria zu ersetzen. Diese Norm trägt die Bezeichnung ISO/IEC 15408. Was also als Alleingang der USA begonnen hat, führte über eine Reihe nationaler Anstrengungen zu international vereinheitlichten Kriterien bis hin zu einem ISO Standard.

1.2.2 Arten

Im Folgenden seien einmal 3 der bekanntesten Vertreter internationaler Sicherheitsstandards näher erläutert.

TCSEC - Die Kriterien des „Orange Books“

Im Orange Book werden sieben Klassen definiert (A1, B3, B2, B1, C2, C1, D). Ein Computersystem, das den Anspruch erhebt, einer dieser Klassen zu entsprechen, muss eine bestimmte Funktionalität und Qualität aufweisen. Das Typische an diesen Kriterien ist, dass eine Klasse auf der anderen aufbaut - hinsichtlich der geforderten Funktionalität (in Bezug auf Sicherheit) als auch hinsichtlich der Qualität der Implementierung. Bei einer höheren Funktionalitätsklasse werden demzufolge auch zusätzliche Anforderungen an den Entwicklungsprozess gestellt. Allerdings sind die Kriterien eher auf Großrechner anwendbar als auf PCs und Workstations, zudem bleiben Netzwerke gänzlich unberücksichtigt. Zusammenfassend kann man sagen, dass man diesen Kriterien ihr Alter und ihre Herkunft „schon ansehen kann“. Das schmälert allerdings nicht den Verdienst ihrer Vorreiterrolle.

ITSEC - Die Kriterien der EU-Länder

In den ITSEC wird von Evaluationsstufen (anstatt Klassen) gesprochen, diese tragen die Bezeichnungen E0 bis E6. Die Stufen geben an, mit welchen Methoden bei der Zertifizierung gearbeitet wird, allerdings sagen sie nichts über die zu prüfende Funktionalität aus. Die Evaluationsstufen sind dabei aufsteigend geordnet. Im Gegensatz zu TCSEC kann der zu überprüfende Funktionsumfang vom Antragsteller beliebig festgelegt werden. Zu diesem Zweck gibt es bereits eine Anzahl vordefinierter Funktionsklassen. Hierbei ist

allerdings zu beachten, dass diese eben nur den Funktionsumfang beschreiben und keine Aussage über die Qualität machen. Daher wird in allen Fällen auch noch die Evaluationsstufe angegeben, welche ein Maß für die Qualität darstellen soll. Zusammenfassend lässt sich sagen, dass ITSEC die Qualität in den Mittelpunkt stellt und die Funktionalität praktisch frei wählbar ist. Im Gegensatz zu TCSEC hat man also eine dreidimensionale Bewertung: Qualität, Funktionalität und Benotung der Funktionalität (niedrig, mittel, hoch).

CC - Die Common Criteria

Die CC weisen große Ähnlichkeiten mit den ITSEC auf. In erster Linie wurde die Flexibilität der Kriterien in Bezug auf ihre Anwendbarkeit erhöht. Die CC gliedern sich in drei Teile. Teil 1 beschreibt ein allgemeines Modell, Teil 2 behandelt funktionale Sicherheitsanforderungen und Teil 3 behandelt Sicherheitsanforderungen zur Vertrauenswürdigkeit. Die CC stellen weiterhin eine Anzahl von vordefinierten Schutzprofilen zur Verfügung, welche die Sicherheitsumgebung, die Sicherheitsziele und auch Sicherheitsanforderungen für Produkttypen (z.B. Firewalls) beschreiben. Hierbei handelt es sich also um verallgemeinerte Sicherheitsvorgaben, losgelöst von jeder konkreten Implementierung. Dabei besteht auch die Möglichkeit eigene Schutzprofile zu definieren. Dafür ist allerdings eine Prüfung notwendig, ob alle Aspekte in sich und untereinander sinnvoll und widerspruchsfrei sind. Die Produktpalette, für die die CC anwendbar sind, ist dabei wesentlich umfangreicher als es bei TCSEC und ITSEC der Fall ist.

1.2.3 Zielgruppen

Die Abbildung 1.1 gibt einen Überblick über die Zielgruppen der verschiedenen Sicherheitskriterien.

1.3 Sicherheitsmodelle

1.3.1 Begriffsklärung

Mit einem Sicherheitsmodell wird versucht, die sicherheitsrelevanten Eigenschaften eines Systems zu beschreiben und einer formalen Analyse zugänglich zu machen.

1.3.2 Modell-Klassifikation

Um ein Sicherheitsmodell zu klassifizieren, müssen als erstes die zu schützenden Einheiten (Objekte) sowie die agierenden Einheiten (Subjekte) des Modells festgelegt werden. Objekte und Subjekte können dabei je nach verwendetem Modell grobkörniger (z.B. Objekte - Dateien , Subjekte - Benutzer) als auch feinkörniger (Objekte - Capability List, Subjekte - Access Control List) modelliert werden. Bei der Festlegung der Zugriffsrechte ist hierbei zwischen universellen und objektspezifischen Rechten zu unterscheiden. Der Zugriff auf die Objekte des Modells kann dabei einer einfachen oder komplexen Zugriffsbeschränkung, wie einer Bedingung über globale Variable (z.B. Zugriff nur während der Bürostunden), unterliegen. Abschließend muss noch die verwendete Sicherheitsstrategie, Zugriffskontroll- oder Informationsfluss-Strategie, betrachtet werden. Bei den Zugriffskontrollstrategien unterscheidet man dabei zwischen benutzerbestimmbaren Zugriffskontrollstrategien (DAC), den systembestimmten (regelbasierten) Festlegungen (MAC) und den rollenbasierten Zugriffskontrollstrategien (RBAC).

Inter- und innerbetriebliche Zielgruppenausrichtung der IT-Sicherheitskriterien Legende: • P ≡ primäre Zielgruppe • S ≡ sekundäre Zielgruppe	IT-Grundschutzhandbuch	ISO 17799 / BS 7799	ISO 13335	ITSEC / Common Criteria	FIPS 140	Task Force-Kataloge	CobiT	Datenschutz-Produktaudit*	ISO 9000
a) Art des Unternehmens									
Hardware-Hersteller	S			S	P	S		P	X
Software-Hersteller	S	S		P	P	P		P	X
Netz-Vermittler		S			S	P	S	S	X
Server-Betreiber	P	P			S	P	S	S	X
Inhalte-Anbieter	P	P				P	S		X
Unternehmen als Anwender	P	P	P		S	S	P		X
b) Rolle innerhalb des Unternehmens									
Management	S	P	P				P	P	P
Projektmanagement	P	P	P	P	P	P	P	P	P
IT-Sicherheitsbeauftragte	P	P	P	P	P	P	S	P	S
IT-Leitung	P	P	P	S	S	P	P	S	S
Administratoren	P	S			S	P	S		S
Revisoren	S	S					P		S

Abbildung 1.1: Zielgruppen von IT-Sicherheitskriterien

1.3.3 Modellarten

Im Folgenden werden einmal die bekanntesten und gebräuchlichsten Sicherheitsmodelle vorgestellt.

Zugriffsmatrix-Modell

Das Zugriffsmatrix-Modell basiert auf der DAC-Strategie. Typische Subjekte sind hierbei Benutzer, Gruppen von Benutzern, Prozesse oder Domains. Typische Objekte sind alle Subjekte („aktive“ Objekte), Speicherbereiche und Dateien, bei Datenbanken die Datenbank selbst, Relationen, Attribute, Sätze oder Felder eines Satzes. Man unterscheidet Zugriffsmatrixmodelle nach statischer bzw. dynamischer Zugriffsmatrix. Die statische Zugriffsmatrix wird bei Anwendungsproblemen eingesetzt, in denen der Rechtezustand a priori bekannt und über längere Zeit konstant ist, z.B. bei einfachen Routern, die als Paketfilter-Firewalls dienen. Ein Eintrag in der Matrix gibt nun an, in welcher Weise ein Subjekt (Zeileneintrag) ein Objekt (Spalteneintrag) benutzen kann. Der gesamte Inhalt der Matrix ist dann der zu einem bestimmten Zeitpunkt entsprechende Zustand des Sicherheitssystems. In der Praxis wird die Matrix jedoch nicht Tabellenform realisiert, da jeweils viele Tabelleneinträge leer wären (Speicherplatzverschwendung). Deshalb erfolgt

eine Beschreibung der Zugriffsrechte durch die beiden Alternativen Capability List (CL), die jedem Subjekt eine Objektliste und Access Control List (ACL), die jedem Objekt eine Subjektliste zuordnet. Die Verwendung einer CL hat hierbei den Vorteil, dass es ziemlich aufwendig ist, alle Subjekte zu ermitteln, die für ein gegebenes Objekt bestimmte Rechte besitzen, während es bei einer ACL ziemlich aufwendig ist, für ein gegebenes Subjekt alle Objekte zu ermitteln, für die das Subjekt bestimmte Zugriffsrechte hat. Ein großer Vorteil der Zugriffsmatrix ist die Flexibilität, denn das Modell bietet die Möglichkeit, Objekte und Subjekte anwendungsspezifisch festzulegen, sowie Zugriffsrechte universell oder objektspezifisch zu modellieren. Damit können Datenintegritätsanforderungen bei entsprechender Modellierung gut erfüllt werden. Die Sicherheitsprobleme einer Zugriffsmatrix ergeben sich bei der CL daraus, dass es relativ einfach festzustellen ist, welche Rechte ein bestimmtes Subjekt hat und im Falle einer ACL die Subjekte, die bestimmte Rechte auf ein gegebenes (passives) Objekt haben, leicht zu ermitteln sind. Größere Mengen von Subjekten, die sich in hoher Frequenz dynamisch ändern, können jedoch nur schlecht mit diesem Modell erfasst werden, da Skalierbarkeit mangelhaft ist.

RBAC-Modell

Das RBAC-Modell basiert auf der RBAC-Strategie und bietet die Möglichkeit, die Berechtigungen zur Nutzung geschützter Komponenten direkt an Rollen und damit an Aufgaben zu knüpfen. Zu diesem Zweck wird festgelegt, welche Subjekte welche Aufgaben durchführen, d.h. in welchen Rollen agieren dürfen. Das RBAC-Modell hat den Vorteil, dass die mangelnde Skalierbarkeit objektbezogener Sicherheitsstrategien stark eingeschränkt wird, da sich die Berechtigungsprofile von Rollen nur selten ändern, d.h. die Verbindung der Rechte zu Rollen relativ stabil sind. Außerdem kann die Rechtevergabe in diesem Modell gemäß des „need-to-know“-Prinzips erfolgen, d.h. es werden nur so viele Rechte vergeben, wie zur Bewältigung der Aufgabe benötigt wird. Die Nachteile des RBAC-Modells ergeben sich aus seiner Komplexität, d.h. es gilt genau abzuwägen, welche Subjekte auftreten, welche Aufgaben diese Subjekte erfüllen sollen und welche Festlegungen hinsichtlich der Zugriffsrechte zu treffen sind. Bei Fehleinschätzungen hinsichtlich der Modellierung kann dies zu gravierenden Sicherheitslücken führen. Allgemein eignet sich das RBAC-Modell für Anwendungsszenarien, in denen Geschäftsprozesse zu erfassen und Berechtigungsprofile zu modellieren sind, die auf organisatorischen, hierarchischen Strukturen basieren. Außerdem eignet sich das Modell aufgrund der guten Skalierbarkeit sehr gut für verteilte Anwendungen sowie IT-Systeme mit sich dynamisch ändernden Subjektmengen, die über einen längeren Zeitraum eine gleich bleibende Menge von Aufgaben im System vollziehen.

Chinese-Wall Modell

Das Chinese-Wall Modell wurde entwickelt, um die unzulässige Ausnutzung von Insiderwissen bei der Abwicklung von Bank- und Börsentransaktionen oder bei der Beratung unterschiedlicher Unternehmen zu verhindern. Die grundlegende Idee des Modells besteht dabei darin, dass die zukünftigen Zugriffsmöglichkeiten eines Subjekts durch die Zugriffe, die es in der Vergangenheit bereits durchgeführt hat, beschränkt werden. Damit soll zum Beispiel verhindert werden, dass ein Unternehmensberater (Subjekt), der Insiderinformationen besitzt, diese Informationen verwendet, um einem Konkurrenzunternehmen Ratschläge zu erteilen. Das Modell basiert auf einem Zugriffsmatrix-Modell, in dem die Menge der Zugriffsrechte explizit durch die universellen Rechte read, write und execute

festgelegt wird. Die zu schützende Objekte des Systems, auf die eine Menge von Subjekten (z.B. Berater) zugreifen kann, werden als Baum strukturiert. Der Schutzzustand des Systems ist hierbei durch zwei Matrizen, eine Matrix für die benutzerbestimmbaren Rechte sowie eine Matrix für die Zugriffshistorie, charakterisiert. Des Weiteren legen zwei systembestimmte Regeln fest, auf welcher Grundlage das Lesen von Objekten sowie das Schreiben von Objekten erlaubt ist. Durch die Leseregel werden "Mauern" errichtet, die verhindern, dass ein Subjekt auf Objekte von verschiedenen Unternehmen, die zur gleichen Interessenskonfliktklasse gehören, zugreifen kann. Die Schreibregel soll zusätzlich gewährleisten, dass kein unerwünschter Informationsfluss durch Schreiboperationen entsteht. Da dem Modell eine Zugriffsmatrix zugrunde liegt, kann die Modellierung von Objekten nicht nur grobkörnig, sondern vorteilhafter feinkörnig, z.B. über ACL's erfolgen. Der wesentliche Nachteil liegt jedoch darin, dass einfache, universelle Zugriffsrechte vorgeschrieben sind und somit z.B. Benutzer oder Prozesse, die diese Rechte nutzen, die entsprechenden zu schützenden Objekte ohne weitere Beschränkungen manipulieren können. Des Weiteren werden die einfachen Zugriffsregeln (DAC-Strategie) über systembestimmte Festlegungen (MAC-Strategie) erweitert, die jedoch sehr restriktiv sind, so dass es nicht möglich ist, über diese Regeln hinausgehend, erlaubte und verbotene Informationskanäle individuell und abhängig von den Anwendungsanforderungen zu modellieren. Das Chinese-Wall Modell ist somit geeignet für Anwendungen, die keine strengen Datenintegritätsanforderungen haben und die keine über die restriktiven Beschränkungen hinausgehenden Vertraulichkeitsanforderungen stellen.

Bell-LaPadula Modell

Dieses Modell basiert auf einem dynamischen Zugriffsmatrix-Modell, wobei die Zugriffsrechte durch die Menge der universellen Rechte read-only, append, execute, read-write, control festgelegt sind. Die Erweiterung des Zugriffsmatrix-Modells erfolgt hierbei durch die Einführung einer geordneten Menge von Sicherheitsklassen (SC), um den zu verarbeitenden Informationen sowie den agierenden Subjekten unterschiedliche Vertraulichkeitsstufen zuzuordnen. Ein Element einer Sicherheitsklasse wird hierbei durch ein Paar X = (A,B) mit A = Sicherheitsmarke und B = Menge von Sicherheitskategorien, beschrieben. Jedem Subjekt s wird eine Sicherheitsklasse (Clearance) und jedem Objekt o eine Sicherheitsklassifikation (Classification) zugeordnet. In diesem Sicherheitsmodell sind zwei systembestimmte Regeln festgelegt. Die erste Regel („no-read-up") garantiert, dass Subjekte niedrigerer Sicherheitsstufe nicht auf Objekte höherer Sicherheitsstufe lesend zugreifen dürfen. Die zweite Regel (no-write-down) legt wiederum fest, dass Subjekte höherer Sicherheitsstufe nicht schreibend auf Objekte niedrigerer Sicherheitsstufe zugreifen dürfen. Vorteilhaft an diesem Modell ist die leichte Überprüfbarkeit und Implementierbarkeit der durch die MAC-Regeln festgelegten Bedingungen. Außerdem wird gewährleistet, dass Informationsflüsse höchstens von unten nach oben entlang der partiellen Ordnung oder innerhalb einer Sicherheitsklasse auftreten können. Nachteilig an diesem Modell ist z.B. die Möglichkeit des blinden Schreibens, d.h. ein Subjekt darf schreibend auf ein höher eingestuftes Objekt zugreifen („no-write-down"-Regel), was im Hinblick auf die Datenintegrität problematisch ist. Das Modell ist aufgrund seiner definierten Festlegungen sehr restriktiv, was zwar hohe Vertraulichkeitsanforderungen (militärischer Bereich) mit einfacher Informationsflussbeschränkung ermöglicht, jedoch den Einsatz begrenzt. Andererseits sind die MAC-Regeln einfach zu überprüfen und effizient zu implementieren, was sich in einer größeren Anzahl kommerzieller Betriebssysteme niederschlägt.

Verbandsmodell

Dieses Modell verallgemeinert den Ansatz vom Bell-LaPadula Modell und beschreibt Informationsflüsse zwischen Datenvariablen eines Programms. Das charakteristische am Verbandsmodell ist der Ansatz, dass die Beschränkungen für Informationsflüsse unabhängig von den Objekten, die die Informationen repräsentieren, festgelegt werden. Zu diesem Zweck werden zulässige und unzulässige Informationskanäle festgelegt, die reglementieren, wie der Umgang mit der Information, die durch Objekte repräsentiert wird, erfolgen soll. Informationen dürfen dabei stets nur innerhalb von Sicherheitsklassen oder gemäß der festgelegten partiellen Ordnung fließen. Ein abwärts gerichteter Informationsfluss (siehe Bell-LaPadula Modell) sowie ein Fluss zwischen Sicherheitsklassen, die nicht miteinander in Beziehung stehen, ist in diesem Modell nicht zulässig. Ein Verbandsmodell kann man sich entsprechend über ein Hasse-Diagramm veranschaulichen. Nachteile ergeben sich daraus, dass für jede Sicherheitsklasse die erlaubten Informationsflüsse erfasst werden müssen, wobei für eine feinkörnige Modellierung von Subjekten entsprechend viele Sicherheitsklassen festzulegen sind. Die Folgen sind hierbei aufwendige, nicht skalierende Verwaltungsmaßnahmen, um differenziert zulässige Informationskanäle festlegen zu können. Ein weiterer Nachteil des Modells liegt in der starren Sicherheitsklassifikation, denn im Wesentlichen erfolgt eine statische Zuordnung zwischen Objekten und ihren Sicherheitsklassen. Das hat meist zur Folge, dass Objekte zu hoch eingestuft sind, um die Anforderungen der Flussrelation erfüllen zu können und niedrig eingestufte Subjekte nur sehr eingeschränkte Rechte erhalten, die in praktischen Anwendungen oft nicht ausreichen. Aus diesem Grund werden im praktischen Einsatz des Verbandsmodells jedoch wieder Subjekte häufig zu hoch eingestuft, es wird also in der Regel gegen das „need-to-know“-Prinzip verstoßen. Allgemein ist das Modell für sehr hohe Vertraulichkeitsanforderungen geeignet, ein Beispiel wäre ein Informationsserver für unterschiedliche Benutzergruppen.

1.4 Verifikation von Sicherheitsmodellen

1.4.1 Begriffsklärung

Mit der Verifikation eines Sicherheitsmodells versucht man die Frage zu klären, ob ein Sicherheitsmodell in seinen spezifizierten funktionalen Sicherheitsmerkmalen (Security Features) mit den nachgewiesenen Sicherheitseigenschaften (Security Properties) übereinstimmt, d.h. die Funktionen leistet, die seiner Spezifikation entsprechen.

1.4.2 Verifikationsmethoden

Testen

Die einfachste Form der Verifikation von Sicherheitsmodellen ist das Testen simulierter Situationen, d.h. man trifft Annahmen über die mögliche Arbeitsumgebung, in der das Modell eingesetzt wird und kreiert entsprechende Testfälle, mit denen man überprüft, ob das System in jeder der simulierten Situationen korrekt reagiert. Da nicht alle Situationen erfasst werden können, ist hierzu eine geeignete Auswahl von zu überprüfenden Situationen zu treffen. Bei dieser Auswahl kann man grundsätzlich drei Fälle unterscheiden. Beim zufälligen Testen („random testing“) erfolgt die Auswahl der simulierten Situationen wie der Name schon andeutet, rein zufällig. Beim strukturellen Testen werden die Testfälle hingegen aus der Programmstruktur abgeleitet, wobei man versucht,

das System gezielt in einen bestimmten Zustand zu versetzen. Diese Zustände können z.B. durch "Hazard"-Analyse bestimmt werden, d.h. durch das Analysieren von nicht erstrebenswerten Ereignissen. Abschließend werden beim funktionalen Testen die einzelnen Aufgaben des Systems getrennt überprüft.

„herkömmliche" Verifikation

Unter herkömmlicher Verifikation versteht man einen nichtformellen Beweis der Korrektheit eines Systems, der durch Annahmen über (Schleifen-)Invarianten oder Zusicherungen ("assertions") geführt wird.

Beweisen

Eine wirklich "sichere" Verifikation eines Modells erlangt man nur durch formales Beweisen, wofür ein formales Sicherheitsmodell notwendig ist. Ein formales Sicherheitsmodell muss entsprechend in einer formalen Notation, mit mathematischen Konzepten für die Syntax und Semantik der Sprache sowie Schlussregeln, erstellt werden. Ein notwendiger Beweis zur Verifikation eines formalen Sicherheitsmodells liegt hierbei in der Überprüfung, ob sich die formal spezifizierten Sicherheitsmerkmale (Security Features) zur Durchsetzung der formal spezifizierten Sicherheitseigenschaften (Security Properties) des Modells eignen. Wird hierbei ein (bekanntes) generisches formales Sicherheitsmodell instanziiert, reicht es aus, zu beweisen, dass die Instanzierung im Rahmen des Modells bleibt, da der Beweis schon induktiv in einem generischen Modell vorhanden ist. Ein Beispiel hierfür ist das Bell-LaPadula Modell, das ein Übergangsschema für Zustandswechsel besitzt, wobei Wechsel nur in jeweils sichere Zustände möglich sind. Dadurch dass der Ausgangszustand in diesem Modell sicher ist, ist die Gesamtsicherheit somit induktiv bewiesen.
Ein weiterer Beweispunkt der Verifikation eines formalen Sicherheitsmodells ist die Widerspruchsfreiheit. In der Regel kann dieser Teil nicht vollständig innerhalb des formalen Kalküls, der für die Spezifikation verwendet wird, behandelt werden. Aus diesem Grund müssen die außerhalb dieses Kalküls liegenden und zum Beweis der Widerspruchsfreiheit notwendigen Konzepte wohlbegründet und auf ein Minimum beschränkt sein. Bei der Unterteilung der formalen Spezifikation in mehrere Ebenen reicht es im Allgemeinen aus, die Widerspruchsfreiheit der untersten Ebene der Spezifikation, d.h. Verifikation der Ebene der Sicherheitsmerkmale, zu beweisen. Wird hierbei ein (bekanntes) generisches formales Sicherheitsmodell instanziiert, so kann der Beweis häufig auf die Widerspruchsfreiheit der konkreten Instanziierung beschränkt werden, vorausgesetzt die Widerspruchsfreiheit des generischen Modells ist gegeben.

1.5 Active Security Suite

Die Active Security Suite fasst die Verwendung eines Sicherheitsmodells, die Integration von Sicherheitspolitiken sowie die Realisierung von Zugriffschutzmechanismen in einer Implementation zusammen. Die Suite wurde 2000 von der Firma "Network Associates" (jetzt McAfee) entwickelt und arbeitet sowohl auf UNIX als auch Windows Systemen. Sie enthält die folgenden Komponenten:

- Gauntlet Proxy-Firewall mit URL-Filter und Virenscanner
- Cybercop-Scanner

- Event Orchestrator
- Net Tools PKI Server

1.5.1 Implementation

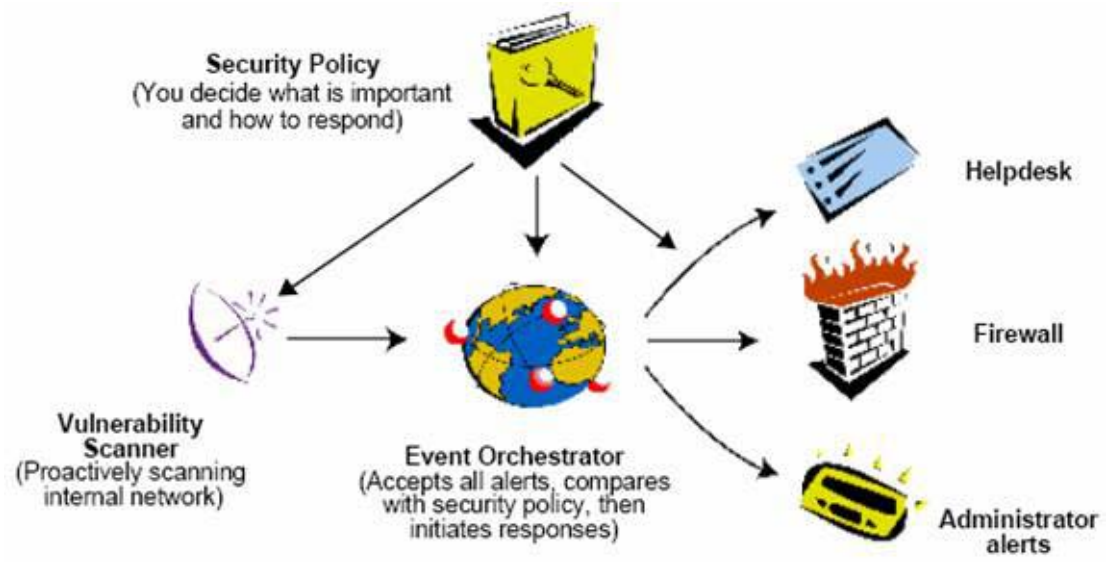


Abbildung 1.2: Implementation der Active Security Suite

Die Abbildung 1.2 soll einmal die Implementation der Active Security Suite verdeutlichen. Hier greifen mehrere Sicherheitskonzepte ineinander und reagieren auf bestimmte Events. Der „Event Orchestrator“ agiert hierbei als zentrale Steuereinheit und entscheidet, anhand von administrativen Vorgaben, die der verwendeten „Security Policy“ entstammen, auf welche Art und Weise auf bestimmte Ereignisse reagiert wird. Dabei überprüft der „Vulnerability Scanner“ regelmäßig das Netzwerk auf Schwachstellen, deren Definitionen ebenfalls der verwendeten „Security Policy“ entstammen. Wird eine Schwachstelle vom Scanner gefunden, wird diese dem „Event Orchestrator“ gemeldet, welcher nun in der „Security Policy“ nach der angemessenen Handlungsweise nachschaut. Ist die Alarmmeldung des Scanners berechtigt, und besteht Handlungsbedarf, ist der „Event Orchestrator“ berechtigt, Prozesse auf laufenden Konsolen zu beenden, sowie der Firewall einen neuen Eintrag hinzuzufügen, sofern ein Angriff von außerhalb des Netzwerkes stattgefunden hat. Ebenso kann der Virenscanner veranlasst werden, bestimmte Rechner auf Viren zu überprüfen, wenn ein Verdacht der Infektion vorliegt. Letztlich wird noch der Administrator des Netzwerkes benachrichtigt, mit ebenfalls (von ihm selbst) vordefinierten Alarm Nachrichten.

1.5.2 Funktionsweise

Anhand von Abbildung 1.3 soll nun die Arbeitsweise der Active Security Suite an 2 Beispielen erklärt werden. Zunächst wird von der Firewall eine Email empfangen und standardmäßig an den Netzwerk-Virenscanner weitergeleitet. Dieser entdeckt einen Virus in der Nachricht und leitet daraufhin eine Warnmeldung über den Virusfund sowie

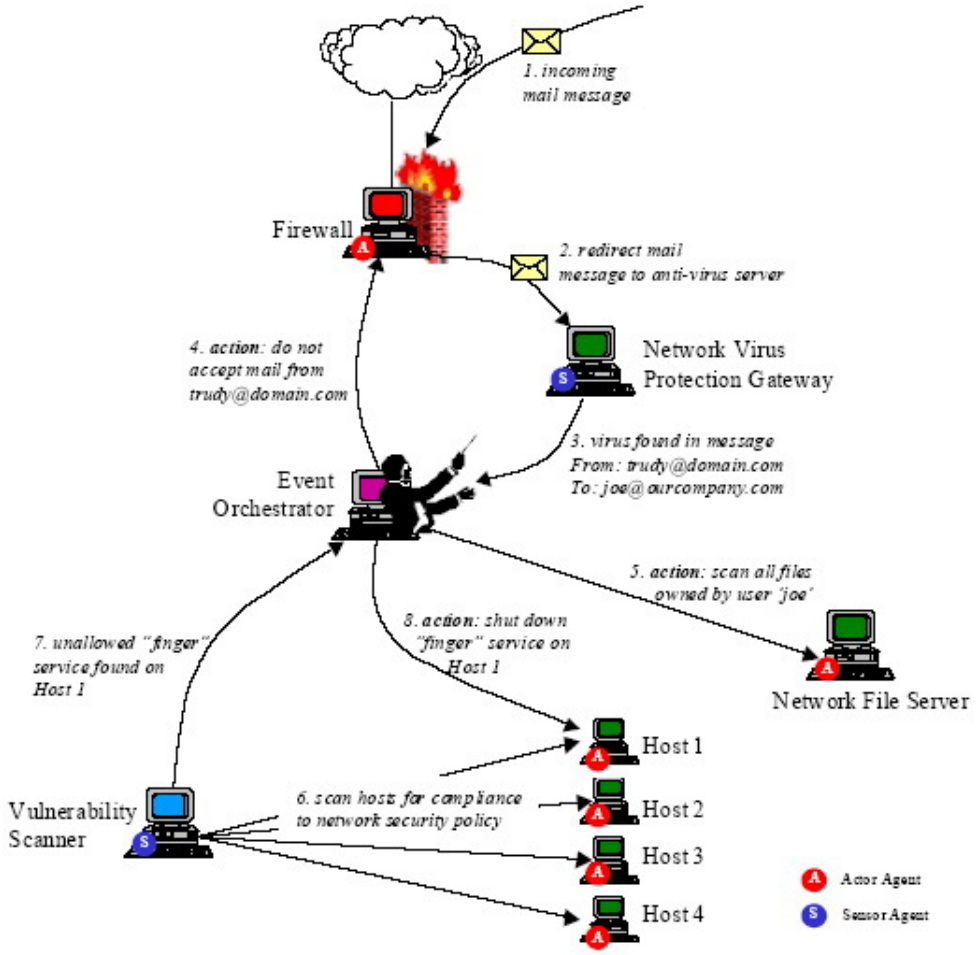


Abbildung 1.3: Arbeitsweise der Active Security Suite

den Absender und Empfänger der Mail an den „Event Orchestrator“ weiter. Dieser entscheidet anhand der aktuell verwendeten Security Policy das zukünftig alle Mails des Absenders geblockt werden. Dazu verschickt er eine entsprechende Nachricht mit der Anweisung an die Firewall, welche ihre Regeln um den gewünschten Eintrag erweitert. Außerdem entscheidet der „Event Orchestrator“ ebenfalls, dass alle vorhandenen Mails des eigentlichen Empfängers auf Viren hin überprüft werden. Dazu wird eine Nachricht an den lokalen „File Server“ mit der entsprechenden Anweisung verschickt. Im zweiten Beispiel entdeckt der „Vulnerability Scanner“ einen unerlaubt laufenden Prozess auf einem Host des Netzwerks. Daraufhin verschickt der Scanner eine Nachricht mit einer Warnmeldung an den „Event Orchestrator“. Dieser entscheidet den Prozess zu beenden und veranlasst selbst den shutdown-Befehl an den entsprechenden Host. Wie man sieht, ist der „Event Orchestrator“ also in der Lage, auf alle Komponenten des Netzwerkes aktiv einzuwirken. Sei es um neue Blockregeln zu erstellen, eine Virenüberprüfung zu veranlassen oder bestimmte Prozesse zu beenden. Allerdings könnte diese Eigenschaft auch missbräuchlich verwendet werden, falls es einem Angreifer gelingen sollte, gefälschte Nachrichten einzuschleusen, die als Befehle des „Event Orchestrators“ angesehen werden. Um dem vorzubeugen, werden alle Nachrichten vor dem Versand signiert. Der Net

Tools PKI Server stellt dafür Zertifikate aus, so dass jede Komponente des Netzwerkes bei dem Empfang einer Nachricht dessen Echtheit überprüfen kann. Außerdem werden alle verschickten Nachrichten innerhalb des Netzwerkes mittels TLS verschlüsselt.

1.5.3 Bewertung

Da die Active Security Suite die Firewall mit dem Virenscanner und dem Content-Scanner kombiniert, ist der Großteil der Integrations- und Konfigurationshürden bereits genommen. Der Administrator schaltet den Virenscanner genauso einfach ein, wie er die Check-Box für die Regeldefinitionen öffnet. Bei anderen Suiten ist dieser Prozess um einiges komplizierter, da dort erst unterschiedliche zu scannende Netzwerk-Objekte definiert werden müssen. Der Event-Orchestrator ist als robustes Event-Managementsystem innerhalb der Suite viel versprechend und sorgt für eine konsequente Umsetzung der Security Policy. Außerdem können bestimmten Ereignissen Schwellenwerte zugewiesen werden, was die Fehlalarme deutlich reduzieren dürfte. Auch die beliebige Verknüpfung von Eventabfolgen, lässt eine komfortable Konfiguration von Ereignissen zu, auf die dann entsprechend reagiert werden kann. Allerdings könnten die Logging- und Reporting-Funktionen umfangreicher sein. Außerdem fehlt eine monitoring-Komponente, mit der man das System zur Laufzeit betrachten kann, komplett. Somit ist die Active Security Suite zwar von Grund auf solide, allerdings ist sie letztendlich auch auf weitere Anwendungen angewiesen, die mit dem „Event Orchestrator“ kommunizieren können. Als Alternative Suiten sind noch die Opsec-Plattform von Check Point und die Etrust-Architektur von CA zu nennen.

1.6 FAZIT

Im Zusammenhang mit IT-Sicherheitsprodukten sind immer zwei Standpunkte zu betrachten: der des Konsumenten und der des Produzenten. Der Konsument möchte eine möglichst klare und verlässliche Aussage über ein Sicherheitsprodukt. Der Produzent möchte hingegen ein möglichst „gutes“ Zertifikat für sein Sicherheitsprodukt mit geringem Aufwand erreichen, um dem Kunden eine gewisse Sicherheit beim Kauf seines Produktes geben zu können. Dabei sollten gerade unabhängige Zertifizierungen wie die Common Criteria und ITSEC an erster Stelle stehen. De Facto ist jedoch jede Zertifizierung mit einem erheblichen Kosten- und Zeitaufwand verbunden. Um IT-Sicherheitsprodukte also „einsetzbar“ zu machen, müssen diese ratifizierbar sein, d.h. große Aussagekraft für den Anwender bei geringem Zusatzaufwand für den Produzenten. Besitzt ein IT-Sicherheitsprodukt nun eine solche Zertifizierung, sagt diese aber immer noch nichts über die Qualität der Umsetzung *der vom Kunden gewünschten Sicherheitspolitik* aus. Sicherheitspolitiken sind dabei nur dann effektiv, wenn sie auch konsequent durchgesetzt werden. Dies ist nur über eine Sicherheitsarchitektur in Form von Sicherheitsmodellen zu erreichen. Hierbei muss das Sicherheitsmodell so konzipiert sein, dass mindestens eine zentrale Komponente die Verwaltung und Durchsetzung der Security Policy für alle eingehenden Anfragen und Überprüfungen übernimmt. Da auch die Hersteller in der Security Branche diesen Sachverhalt erkannt haben, erscheinen immer mehr Gesamtsicherheitskonzepte in Form von „Security Suiten“ auf dem Markt. Diese sollen, zumindest laut Hersteller, einen rundum-Schutz in jedem Netzwerk liefern. Da nun aber nicht jeder Hersteller gleich versiert in den Bereichen Firewall, Virenschutz, Vulnerability Scanning und Event-Management ist, werden bestehende Lücken der Suite kurzerhand durch

Fremdkomponenten geschlossen. Dabei ergeben sich allerdings gleich mehrere Probleme. Zunächst ist es nie leicht Fremdkomponenten in ein bestehendes System zu Integrieren, Fehlerquellen sind nicht ausgeschlossen. Weiterhin ist der Kunde quasi gezwungen die Komponenten in Kauf zu nehmen, welche in der Security Suite enthalten sind, ohne das dieser die Möglichkeit hat, sich die Komponenten entsprechend seiner Anforderungen auszusuchen. Ist dieser Sachverhalt einmal überwunden und das Produkt an den Kunden gebracht, äußert sich das nächste Problem im Support. Der ursprüngliche Hersteller der eingebundenen Komponente ist dafür nicht zuständig, da er sein Produkt nicht direkt an den Kunden sondern an eine andere Firma verkauft hat. Die Firma, welche die Fremdkomponenten in Ihr System integriert hat, kann aber in den seltensten Fällen das gleiche Know-how aufbringen, wie es der Original Hersteller kann. Sofern zwischen diesen Firmen ein Support Vertrag besteht, kann der Kunde zumindest auf eine Lösung für sein Problem hoffen. Ein möglicher Ansatz für eine kundenfreundlichere Lösung, den einige Hersteller auch bereits gehen, ist seine eigenen Komponenten für die Verwendung in anderen Architekturen zu zertifizieren um somit die Kompatibilität zu gewährleisten. Somit hat der Kunde auch Anspruch auf den Support vom Original Hersteller, sofern dessen Komponenten in der verwendeten Security Suite integriert sind. Ob sich allerdings diese (Fremd-)Zertifizierungen, die ebenfalls mit nicht grade geringen Kosten verbunden sind, im Softwarebereich und speziell im Sicherheitsbereich durchsetzen, bleibt abzuwarten.

Literaturverzeichnis

[1] Network Associates. Active Security - Getting Started Guide. http://commerce.softbank.co.kr/html/tech/Tech_File_bbs/data/Active_Security_Getting_Started_Guide.pdf, März 1999.

[2] [ECKERT] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, 2 edition, 2001.

[3] Uni Essen. Sicherheitsmodelle. http://dawis.informatik.uni-essen.de/site/teaching/ss2004/siap/VlMaterial/Sicherheitsmodell-Grundlagen.pdf, Mai 2004.

[4] Bundesamt für Sicherheit in der Informationstechnik. IT-Sicherheitskriterien. http://www.bsi.de/zertifiz/itkrit/itkrit.htm.

[5] Bundesamt für Sicherheit in der Informationstechnik. Leitfaden für die Erstellung und Prüfung formaler Sicherheitsmodelle im Rahmen von ITSEC und Common Criteria. http://www.aktionpuppe.de/studium/Hauptseminar_NI/leitfaden.pdf, Jan 2003.

[6] F.Tietz H.Scheelken. IT-Sicherheit im Internet. http://www.zedat.fu-berlin.de/schulung/pdf/IT-Sicherheit.pdf, Jan 2003.

[7] Alexander Osherenko. Noninterference-Sicherheitsmodell. http://www.informatik.hu-berlin.de/Institut/struktur/algorithmenII/Lehre/WS2003-2004/Sem_Security/11NonInterference/Noninterference.pdf, Feb 2004.

[8] Dr. Jörg Pflüger. Sicherheitsmodellierung und Zugriffskontrollsysteme. http://igw.tuwien.ac.at/igw/lehre/Verlaesslichkeit_DuD/DuD2001_skripte/DsDs2001Skript7.pdf, 2002.

[9] Arbeitsgruppe: Sicherheit und Vertrauen im Internet. IT-Sicherheitskriterien im Vergleich. http://www.initiatived21.de/druck/news/publikationen2002/doc/22_1053502380.pdf, Dez 2001.

2 Windows 2000: Mechanismen der Durchsetzung und Verwaltung von Rechten und Politiken

Daniel Kaulbars, Jörn Hartwig

Abstract

Das Ziel von Sicherheit ist unberechtigte Zugriffe zu verhindern, die in einer Umgebung stattfinden, in der mehrere Benutzer Zugriff auf dieselben physischen Ressourcen oder Netzwerkressourcen haben. Das Betriebssystem und die einzelnen Benutzer müssen daher die Möglichkeit haben, Dateien, Speicher und Konfigurationseinstellungen vor unbefugter Einsichtnahme und unerwünschten änderungen zu schützen. Zu den Sicherheitsfunktionen des Betriebssystems gehören Mechanismen, die das Betriebssystem vor Beschädigungen schützen, die Benutzer mit sehr eingeschränkten Rechten daran hindern, bestimmte Operationen auszuführen; und die Benutzeranwendungen daran hindern, die Programme anderer Benutzer oder das Betriebsystem nachteilig zu beeinflussen.

2.1 Einführung

Das Paper beschäftigt sich mit dem Thema der Realisierung von Rechten und Politiken in Windows 2000. Es wird erläutert, wie das Designs und die Implementierung von Microsoft Windows 2000 durch die strikten Anforderungen, ein robustes Sicherheitssystems zu implementieren, beeinflusst wurden.
Policies unter Windows beschreiben Ressourcen, die für Einzelpersonen oder Personengruppen verfügbar sind bzw. ihnen verwehrt bleiben sollen. Im Kapitel Group Policies, werden die Möglichkeiten und Konzepte der einzelnen Policies unter Windows 2000 vorgestellt. Neuerungen und Veränderungen zu Windows NT 3.5 werden aufgezeigt und deren Auswirkungen verdeutlicht.
Die dann folgenden Abschnitte werden zeigen, wie diese, bereits genannten Policies, in Windows 2000 ihre Umsetzung finden und auf welche, durch das System bereitgestellte, Mechanismen und grundlegenden Konzepte sie zurückgreifen. Es werden im Kapitel SSicherheitseinstufungen"die sieben Sicherheitsstufen des Orange Books, bzw. das Orange Book selbst, vorgestellt. Weiterhin wird versucht werden, die These, dass Windows 2000, ähnlich wie Windows NT 4 (SP 6a), der Sicherheitsstufe C2 entspricht, zu belegen. Zur Veranschaulichung dieser These werden einige ausgewählte Komponenten, die die Sicherheitsaspekte in Windows 2000 realisieren, vorgestellt um verknüpfend zu zeigen, warum und vor allem wie, diese Komponenten unter Zuhilfenahme bestimmter Mechanismen, die Sicherheitsanforderungen der Stufe C2 umsetzen.

2.2 Group Policies

Die in Windows NT 4.0 verwendeten System- und Sicherheitsrichtlinien wurden in Windows 2000 durch ein ein Konzept von Gruppenrichtlinien, auch Group Policies genannt, ersetzt. Im folgenden Kapitel werden Group Policies und deren prinzipielle Struktur beschrieben sowie Unterscheide und Gemeinsamkeiten im Vergleich zu Windows NT typischen System- und Sicherheitsrichtlinien betrachtet.

2.2.1 Was sind Group Policies

Gruppenrichtlinien definieren zentral alle Konfigurationseinstellungen für einen jeweiligen Benutzer bzw. Computer. Diese Definition besitzt ihre Gültigkeit innerhalb eines Standortes, einer Domäne oder einer anderen Organisationseinheit. Group Policies definieren bestimmte Aspekte der Benutzerumgebung, spezifizieren das Systemverhalten und schränken die Rechte eines jeweiligen Benutzers ein. Eine Policy ist also eine mit einer bestimmten Gruppe in Verbindung stehende Einstellung, die durch einen Administrator verändert bzw. generell gesetzt werden kann. Viele dieser Einstellungen kommen in den Schlüsseln der Registrierung zum Einsatz. Wie diese Verwendung im Detail erfolgt, soll hier aber nicht weiter beschrieben werden. Es wird sich lediglich auf Grundmechanismen und damit verbundene Auswirkungen beschränkt.

2.2.2 Unterschiede von NT 4.0 zu Windows 2000

Die Gruppenrichtlinien unter Windows 2000 bringen im Vergleich zu den System- und Sicherheitsrichtlinien für Windows NT 4.0 einen deutlich erweiterten und verbesserten Leistungsumfang mit sich.

System Policies unter NT 4.0

Die Windows NT Policies werden über den Systemrichtlinieneditor verwaltet und aktualisiert und gelten nur auf der jeweiligen Domänenebene. Mitglieder, die einer bestimmten Gruppe angehören sind berechtigt diese zu deaktivieren bzw. zu aktivieren. Unter speziellen Voraussetzungen ist es dem Benutzer möglich unter Zuhilfenahme des Registrierungseditors diese Richtlinien zu manipulieren.
Alle Richtlinien von Windows NT 4.0 sind fest in der Registrierung verankert und bleiben bis zu einer expliziten überschreibung in der Registrierung erhalten. Diese Beständigkeit könnte dazu führen, dass Registrierungseinstellungen und Policies erhalten bleiben, auch wenn sich die Zugehörigkeit eines Nutzers zu Sicherheitsgruppen geändert hat.
Alle Richtlinien beschränken sich im Wesentlichen auf die Gestaltung der Desktopumgebung.

Group Policies unter Windows 2000

Unter Windows 2000 sind Policies das zentrale Werkzeug zur Änderungs - und Konfigurationsverwaltung von Policies sind speziellen Standorten, Domänen oder Organisationseinheiten zugeordnet und gelten nur für die Benutzer innerhalb eines Standorts, einer Domäne oder Organisationseinheit und allen jeweils untergeordneten Strukturen. Alle dieser Gruppenrichtlinien können, durch die Mitgliedschaft in bestimmten Sicherheitsgruppen, von Computern oder Personen deaktiviert bzw. aktiviert werden. Sämtliche Einstellungen können nur vom Administrator geändert werden, es besteht also nicht wie

in Windows NT 4.0 die Möglichkeit der änderung von Policies durch den Nutzer selbst. Bei einer Veränderung der Gruppenzugehörigkeit eines Nutzers, werden die entsprechenden Einträge sofort aus der Registrierung entfernt sobald ein Gruppenrichtlinienobjekt nicht mehr zutreffend ist. Die Funktion des Gruppenrichtlinienobjekts wird im nächsten Unterpunkt Group Policies und Active Directory genauer erläutert werden. Die Gruppenrichtlinien von Windows 2000 bieten im Vergleich zu den Policies von Windows NT 4.0 weitaus mehr Möglichkeiten zur Gestaltung der Desktopumgebung sowie zahlreiche Gestaltungsmöglichkeiten der gesamten Netzwerkumgebung eines Benutzers.

2.2.3 Group Policies und Active Directory

Alle Gruppenrichtlinien sind in lokalen, und je nach Bedarf, nichtlokalen Gruppenrichtlinienobjekten gespeichert. Unterscheidungskriterium ist hierbei der Speicherort. Auf jedem Computer, der unter Windows 2000 läuft, ist ein Gruppenrichtlinienobjekt vorhanden. Nichtlokale Gruppenrichtlinien sind Teil des Active Directory (Verw. 1) und mit Standorten, Domänen oder Organisationseinheiten verknüpft. Die Gruppenrichtlinienobjekte sind als Objekte im Active Directory mit in die Active Directory Replikation eingebunden.

2.2.4 Vererbung von Group Policies

Gruppenrichtlinien wirken sich auf alle Container innerhalb des Active Directorys aus, mit denen das entsprechende Gruppenrichtlinienobjekt verknüpft ist. Sie sind kumulativ und werden an darunterliegende Container vererbt. Es können mehrere Container mit einem Gruppenrichtlinienobjekt verbunden sein. Gleiches gilt für die Verbindung mehrerer Gruppenrichtlinienobjekte mit einem Container. Die Vererbung erfolgt in der Reihenfolge Standort, Domäne und Organisationseinheit. Verfügen sowohl die übergeordnete wie auch die untergeordnete Organisationseinheit über Gruppenrichtlinien die kompatibel sind, so werden beide angewendet. Sind die Gruppenrichtlinien nicht kompatibel, wird für die untergeordnete Organisationseinheit auch nur die ihr direkt zugeordneten Richtlinie wirksam. Die Richtlinie der übergeordneten Organisationseinheit wird folgerichtig nicht vererbt. Die Vererbung lässt sich durch verschiedene Einstellungen beeinflussen. Ebenso ist es möglich den dargestellten Blockierungsmechanismus einer untergeordneten Organisationseinheit außer Kraft zu setzten, oder die Richtlinienvererbung der übergeordneten Organisationseinheit zu deaktivieren.

2.2.5 Berechtigungen auf Group Policies

Nachdem sich die Anwendung der Gruppenrichtlinien über die Vererbung nur auf der Ebene von Containern steuern lässt, erfolgt eine Art Feinsteuerung über die Berechtigungsvergabe auf das entsprechende Gruppenrichtlinienobjekt. Die Richtlinien wirken sich nur auf Benutzer oder Computer aus, die für dieses Gruppenrichtlinienobjekt die Leseberechtigungen und Gruppenrichtlinien übernehmen verfügen. Die Filterung erfolgt dann, indem nur bestimmten Gruppen von Benutzern oder Computern die Leseberechtigungen und Gruppenrichtlinien übernehmen erteilt werden oder aber ausgewählten Gruppen diese Berechtigungen explizit verweigert werden. Ein weiteres Anwendungsgebiet der Berechtigungen kann die Delegation der Richtlinienverwaltung von den Domänen-Administratoren auf andere Gruppen sein, indem diesen die Berechtigungen Lesen und Schreiben zugewiesen werden.

2.2.6 Anwendungsreihenfolge der Group Policies

Als erstes wird das lokale Gruppenrichtlinienobjekt angewandt. Danach das Gruppenrichtlinienobjekt auf Standort- bzw. Domänenebene und als letztes auf Ebene der Organisationseinheit. Diese Reihenfolge spielt insbesondere dann eine Rolle, wenn Einstellungen überschrieben werden. Hierbei kommt grundsätzlich die zuletzt angewendete Einstellung zur Anwendung. Einstellungen von hierarchisch übergeordneten Gruppenrichtlinienobjekten können also grundsätzlich überschrieben werden, soweit dies nicht durch eine spezielle Konfiguration der Vererbung verhindert wird.

2.2.7 Einsatzmöglichkeiten der Group Policies

Als Verwaltungswerkzeug der Gruppenrichtlinien unter Windows 2000 wird der unter Windows NT 4.0 eingeführte Systemrichtlinieneditor durch das Gruppenrichtlinien-Snap-In für die Microsoft Management Konsole ersetzt. über das Snap-In lassen sich innerhalb der zwei Hauptgruppen, Computerkonfiguration und Benutzerkonfiguration, die folgenden Einstellungen vornehmen.

Scripte

Darüber können Skripts angesprochen werden die beim An- und Abmeldevorgang (Benutzer) bzw. Starten und Herunterfahren (Computer) ausgeführt werden. Es können alle vom Windows Scripting Host unterstützten Sprachen (VBScript, JavaSkript, Perl und MS-DOS) verwendet werden.

Softwareinstallation

über das Menü Softwareinstallation können Anwendungen Gruppen von Benutzern oder Computern zugeordnet und installiert werden. Hierfür ist es erforderlich, dass die Clients unter Windows 2000 laufen und über eine spezielle Erweiterung für die Softwareinstallation verfügen.

Verzeichnisumleitungen

über die Verzeichnisumleitung können Zugriffe auf ursprünglich lokale Ordner wie z.B. Eigene Dateien, Desktop, Startmenü etc. ins Netzwerk umgeleitet werden, um den zentralen Zugriff zu ermöglichen.

ADM-Dateien

Administrative Vorlagen enthalten registrierungsbasierte Richtlinien, die das Verhalten und die Erscheinung des Desktops einschließlich der Komponenten und Anwendungen des Betriebssystems bestimmen.

Sicherheitseinstellungen

Von zentraler Bedeutung für die gesamte Netzwerksicherheit sind die Sicherheitseinstellungen in den Gruppenrichtlinien.

Kontenrichtlinien
enthalten Richtlinien für Kennwörter (z.B. Länge, Gültigkeit etc.), Kontosperre

(z.B. Anzahl Anmeldeversuche, Dauer der Sperre) und zur Kerberos-Authentifizierung.

Richtlinien für lokale Computer
enthalten Richtlinien für die überwachung von sicherheitsrelevanten Ereignissen (Anmeldungen, Dateizugriffe, etc.), Zuweisung von Benutzerrechten (lokale Anmeldung, Zugriff über das Netzwerk, Inbesitznehmen von Dateien etc.) und für Sicherheitsoptionen (z.B. Abmeldung erzwingen, Smartcard-Anmeldung erzwingen etc.)

Richtlinien für Ereignisprotokolle
enthalten Einstellungen wie Größe und Aufbewahrungsdauer für die Protokollierung auf Anwendungs-, System und Sicherheitsebene.

Richtlinien für eingeschränkte Gruppen
legen fest, welche Benutzer zu den Gruppen mit besonderen Berechtigungen gehören. Hierzu gehören die vordefinierten Standardgruppen sowohl auf globaler Ebene (z.B. Domänenandministratoren) wie auch auf lokaler Ebene (z.B. Administratoren, Hauptbenutzer, Druckoperatoren und Serveroperatoren). Auch benutzerdefinierte Gruppen können zu den eingeschränkten Gruppen hinzugefügt werden. Hierdurch wird die Zugehörigkeit zu diesen Gruppen erzwungen und es werden keine lokalen Variationen zugelassen. So werden z.B. änderungen der Gruppenzugehörigkeit durch den lokalen Administrator überschrieben.

Richtlinien für Systemdienste
bestimmen den Startmodus von Systemdiensten und die Zugriffsberechtigungen auf Systemdienste. Sie legen Rechte und Berechtigungen fest mit denen Systemdienste ausgeführt werden und erweitern die überwachungsmöglichkeiten für Systemdienste.

Registrierungsrichtlinien
definieren die Sicherheitseinstellungen für Registrierungsschlüssel. Hierunter fallen insbesondere die Zugriffsrechte, die überwachung von Zugriffen sowie die Besitzerrechte.

Dateisystemrichtlinien
bestimmen die Sicherheitseinstellungen auf Dateien, hauptsächlich die Zugriffsberechtigungen, die überwachung und die Besitzerrechte

Richtlinien für öffentliche Schlüssel
fügen zusätzliche Wiederherstellungsagenten (für die Wiederherstellung EFS-verschlüsselter Daten) hinzu, die automatische Registrierung und Erneuerung von Computerzertifikaten festgelegen sowie Listen der vertrauenswürdigen Zertifizierungsstellen verwalten.

IP-Sicherheitsrichtlinien im Active Directory
legen fest, welche Authentifizierungs- und Kommunikationsstandards mit IP-Clients und IPSec-fähigen Clients verlangt werden (z.B. nur verschlüsselte Kommunikation

zulassen, Verhindern der Kommunikation mit nicht IPSec-fähigen Clients).

2.2.8 Remoteinstallationsdienste

Remoteinstallationsdienste sind eine optionale Komponente unter Windows 2000 Server, welche zur Remoteinstallation von Betriebssystemen genutzt werden. über die Gruppenrichtlinien kann festgelegt werden, welche Auswahlmöglichkeiten der Benutzer im Clientinstallationsassistenten zur Verfügung hat. Die Gruppenrichtlinien unter Windows 2000 stellen ein effektives Instrument zur Clientverwaltung dar. Gegenüber den Systemrichtlinien unter Windows NT 4.0 wurde der Leistungsumfang deutlich erweitert. Durch den zielgerichteten Einsatz der Gruppenrichtlinien können deutliche Verbesserungen hinsichtlich Benutzerproduktivität und Gesamtbetriebskosten erzielt werden. Dies setzt jedoch eine sorgfältige Planung und ein abgestimmtes Konzept voraus, das auch die Effekte der Vererbung und der Berechtigungen auf Gruppenrichtlinien mit berücksichtigt.

2.3 Sicherheitseinstufungen

Dieses Kapitel beschäftigt sich mit den Einflüssen von Sicherheitskriterien, die bei dem Design von sicheren Systemen berücksichtigt werden müssen. Das National Computer Security Center (NCSC) wurde 1981 als Abteilung der National Security Agency (NSA) des US-Verteidigungsministeriums gegründet, um Regierung, Unternehmen und Verbraucher darin zu unterstützen, in Computersystemen gespeicherte Daten zu schützen. Im Rahmen dieser Aufgabenstellung definierte die NCSC Bereiche von Sicherheitseinstufungen (siehe Tabelle 1), die eingesetzt werden, um anzugeben, in welchem Umfang kommerzielle Betriebssysteme, Netzwerkkomponenten und vertrauenswürdige Anwendungen Schutz bieten. Diese Sicherheitseinstufungen, die anhand der Trusted Computer System Evaluation Criteria (TCSEC) des US-Verteidigungsministeriums festgelegt wurden, wurden 1983 verabschiedet und sind im englischsprachigen Raum unter der Bezeichnung Örange Book"bekannt. Im Juli 1995 wurde Microsoft Windows NT 3.5 mit SP 3 als erste

Einstufung	Beschreibung
A1	Geprüftes Design
B3	Sicherheitsdomänen
B2	Strukturierter Schutz
B1	Ausgewiesener Sicherheitsschutz
C2	Kontrollierter Zugriffsschutz
C1	Diskreter Zugriffsschutz
D	Minimaler Schutz

Tabelle 2.1: TCSEC-Sicherheitseinstufungen

Version von Windows NT bewertet. Sie entspricht den Anforderungen der C2-Einstufung. Im März 1999 bekam Windows NT 4 mit SP 3 von der britischen Behörde IETSEC (Information Technology Security) die E3-Sicherheitsstufe zugesprochen. Diese Einstufung ist der amerikanischen C2-Einstufung äquivalent.

Windows NT 4 mit SP 6a wurde im November 1999 schließlich auch mit der C2-Einstufung angesetzt. Welche Kriterien für eine C2-Einstufung erforderlich sind, werden nachfolgend kurz aufgeführt.

1. Eine sichere Anmeldefunktion, die eine eindeutige Identifizierung der Benutzer erfordert. Dem Benutzer wird nur dann Zugriff auf den Computer gewährt, nach dem dieser sich auf irgendeine Weise authentifiziert hat.

2. Eine Zugriffssteuerung, die es jedem Besitzer ermöglicht festzulegen, welcher Nutzer Zugriff auf seine Ressource erhält und welche Operationen Benutzer auf der Ressource ausführen dürfen. Der Besitzer gewährt Rechte, die einer Gruppe von Nutzern bestimmte Arten des Zugriffs (lesen, schreiben, ausführen) ermöglichen.

3. Eine Sicherheitsüberwachung, die es ermöglicht, alle sicherheitsrelevanten Ereignisse sowie alle Versuche, auf Systemressourcen zuzugreifen, sie zu erstellen oder zu löschen, zu erkennen und aufzuzeichnen. Mit Hilfe der Anmeldekennung werden die Identitäten aller Benutzer verzeichnet, so dass sich einfach nachverfolgen lässt, wer welche Operation auszuführen versuchte.

4. Ein Schutz hinsichtlich der Objektwiederverwendung, der Benutzer daran hindert, auf Daten zuzugreifen, die von einem anderen Benutzer gelöscht wurden, oder auf Speicher zuzugreifen, der zuvor von einem anderen Benutzer verwendet und freigegeben wurde. Zur Gewährleistung müssen sämtliche Objekte, einschließlich Dateien und Speicher initialisiert werden, bevor sie einem Benutzer zugewiesen werden.

Windows NT erfüllt zudem folgende Anforderungen der Sicherheitsstufe B.

1. Funktionen für vertrauenswürdige Pfade, die Programme des Typs Trojanisches Pferd daran hindern, Benutzernamen und Kennwort während der Benutzanmeldung abzufangen. In Windows NT wird diese Funktionalität durch die Anmeldesequenz Strg + Alt + Entf umgesetzt. Diese Tastenkombination, auch als SAS Sequenz (Secure Attention Sequence) bezeichnet, führt stets zur Anzeige des Anmeldedialogfeldes. Auf diese Art und Weise können Trojanische Pferde rechtzeitig und eindeutig erkannt werden.

2. Verwaltung vertrauenswürdiger Funktionen, die realisiert, dass getrennte Benutzerkonten mit verschiedenen Verwaltungsfunktionen eingerichtet werden können. Beispielsweise stehen getrennte Konten für Administratoren, für Benutzer, die für die Datensicherung des Computers verantwortlich sind, und Standardbenutzer zur Verfügung.

Auf Grund dessen, dass sich der Evaluierungsprozess über mehrere Jahre erstreckt, wird es wohl noch einige Zeit dauern, bis die Einstufung von Windows 2000 abgeschlossen ist. Das grundlegende Sicherheitsmodell von Windows 2000 ist jedoch eine robuste Weiterentwicklung des Modells von Windows NT 4. In den nächsten Abschnitten soll beschrieben werden, dass die Entwickler von Windows 2000 berechtigter Weise Ambitionen anmelden, dass Windows 2000 in der C2-Sicherheitseinstufung einsortiert wird. Außerdem wird Aufschluss darüber gegeben in wie weit die Sicherheitskriterien Einfluss auf das Design des Sicherheitsmodells von Windows 2000 genommen haben.

2.4 Das Sicherheitssystem von Windows 2000

Als Grundlage für das Unterkapitel 4.2 wird in dem folgenden Abschnitt beschrieben, welche Komponenten und Datenbanken von Windows 2000 involviert sind, um das Sicherheitssystem zu implementieren.

2.4.1 Die Architektur des Sicherheitssystem

Ein Gesamtüberblick über die Architektur des Sicherheitssystems gibt die aufgeführte Abbildung 1. Aufgezeigt werden involvierten Akteure und Speicher des Benutzer- bzw. des Kernelmodus der Systemarchitektur von Windows 2000.

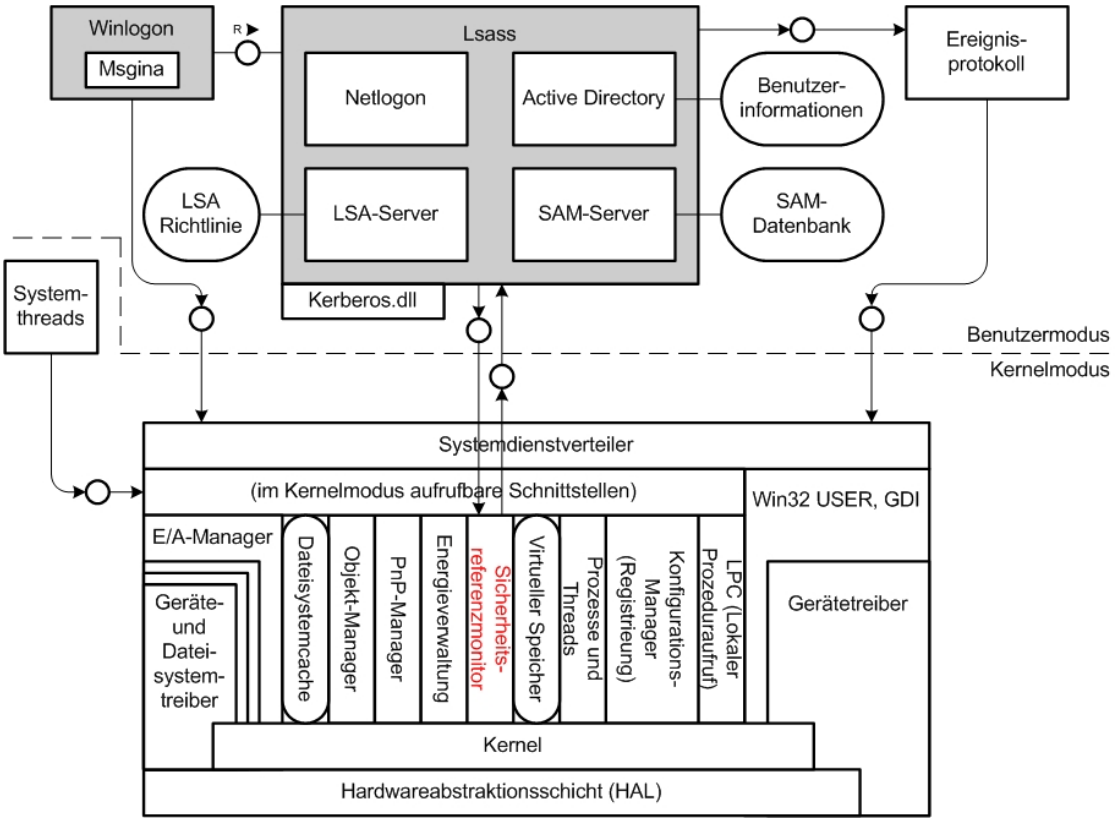


Abbildung 2.1: Beziehungen und Komponenten des Sicherheitssystems von Windows 2000

1. **Sicherheitsreferenzmonitor (SRM)** Der SRM ist eine Komponente (\Winnt\System32\Ntoskrnl.exe) die in der Windows 2000-Ausführungsschicht liegt und dafür zuständig ist, die Berechtigung zum Zugriff auf Objekte zu prüfen, Benutzerrechte zu überwachen und entsprechende Sicherheitsüberwachungsnachrichten auszugeben.

2. **Teilsystem der lokalen Sicherheitsinstanz (Lsass)** Lsass ist zuständig für für die lokalen Sicherheitsrichtlinien, die Benutzerauthentifizierung und das Senden von Sicherheitsüberprüfungsnachrichten an das Ereignisprotokoll. Der Benutzermodusprozess führt die Binärdatei (\Winnt\System32\Lsass.exe) aus. Der lokale Sicherheitsauthentifizierungsdienst (Lsasrv - \Winnt\System32\Lsasrv.dll), eine Bibliothek, die von Lsass geladen wird, implementiert den Großteil dieser Funktionalität.

3. **Lsass-Richtliniendatenbank** Die Einstellungen für die lokalen Systemrichtlinien sind in dieser Datenbank enthalten. Sie wird in der Registrierung unter dem Schlüssel HKLM\SECURITY gespeichert. Sie beinhaltet Informationen, wie z. B. welche Domänen Anmeldeversuche authentifizieren dürfen, wer Zugriff auf das System hat, wie dieser Zugriff erfolgen kann, wem welche Berechtigungen erteilt werden und welche Art der Sicherheitsüberprüfung ausgeführt werden soll. Zusätzlich werden unter dem Schlüssel Secrets auch "geheimeInformationen gespeichert, wie die Anmeldedaten, die für zwischengespeicherte Domänenanmeldungen und Anmeldungen von Benutzerkonten als Win32-Dienst verwendet werden.

4. **Sicherheitskontenverwaltung (SAM)** Hier werden Subroutinen implementiert, die für die Verwaltung der Datenbank verantwortlich sind, in der die auf dem lokalen Rechner definierten Benutzer und Gruppen gespeichert sind. Ausgeführt wird der Security Accounts Manager, der in der Bibliothek \Winnt\System32\Samsrv.dll implementiert ist, im Lsass-Prozess.

5. **SAM-Datenbank** SAM realisiert die Datenbank, in der die definierten lokalen Benutzer und Gruppen sowie deren Kennwörter und anderen Attribute abgelegt werden. Sie ist in der Registrierung unter dem Schlüssel HKLM\SAM gespeichert.

6. **Active Directory** Active Directory ist ein Verzeichnisdienst, der eine Datenbank umfasst, die Informationen zu Objekten einer Domäne enthält. Es werden Informationen über die Objekte (Benutzer, Gruppen oder Computer) gespeichert. Der Active Directory-Server, der in der Bibliothek \Winnt\System32\Ntsda.dll implementiert ist, wird ebenfalls im Lsass-Prozess ausgeführt.

7. **Authentifizierungspakete (Kerberos.dll, Msv1_0.dll)** sind DLL's, die im Kontext des Lsass-Prozesses ausgeführt werden und die Authentifizierungsrichtlinien von Windows 2000 implementieren. Sie überprüfen, ob ein gegebener Benutzername und ein Kennwort zusammengehören, und falls dem so ist, werden an Lsass Informationen zum Sicherheitsprofil des Benutzers zurückgegeben.

8. **Anmeldeprozess (Winlogon)** Der Anmeldeprozess wird durch eine Benutzermodus-DLL umgesetzt, die die Datei \Winnt\System32\Winlogon.exe ausführt. Diese Datei ist dafür zuständig, auf die Anmeldetastenkombination (SAS) zu reagieren und die interaktiven Anmeldesitzungen zu verwalten. Winlogon erstellt während der Anmeldung die Benutzeroberfläche für eine Sitzung.

9. **Grafische Identifikation und Authentifizierung (GINA)** ist ebenfalls eine Benutzermodus-DLL, die im Winlogon-Prozess ausgeführt wird. Mit Hilfe von Winlogon wird der Name und das Kennwort oder die Smartcard-PIN eines Benutzers ermittelt. Die Standard-GINA von Windows 2000 befindet sich in der \Winnt\System32\Msgina.dll.

10. **Netzwerkanmeldedienst (Netlogon)** Ein Win32-Dienst (\Winnt\System32\Netlogon.dll), der im Lsass-Prozess ausgeführt wird und auf Netzwerkanmeldeanforderungen reagiert. Die Authentifizierung wird ebenso wie bei lokalen Anmeldungen gehandhabt, in dem der Benutzername und das Kennwort zur Überprüfung an Lsass gesandt werden.

11. **Kernelsicherheitsgerätetreiber (KSecDD)** Der Kernel Security Device Diver ist eine Kernelmodusbibliothek mit Funktionen, die die LPC-Schnittstellen (LPC - Local Procedure Call) implementieren, die andere im Kernelmodus laufende Sicherheitskomponenten, einschließlich des EFS-System (EFS - Encrypting File System), verwenden, um mit Lsass zu kommunizieren. KSecDD befindet sich in \Winnt\System32\Drivers\KSecDD.sys.

Die Kommunikation zwischen Usermodus und Kernelmodus wird über zwei Kanäle zwischen dem Lsass-Prozess und dem SRM sichergestellt. Diese zwei Kanäle werden mit Hilfe von zwei Ports realisiert. Während der Systeminitialisierung erstellt der SRM einen Anschluss namens SeRmCommandPort, zu dem Lsass eine Verbindung herstellt. Der Lsass-Prozess erstellt ebenfalls bei seinem Start einen Anschluss, den SeLsaCommandPort, zu dem der SRM eine Verbindung herstellt. Sind beide Verbindungen initialisiert, hören die Komponenten auf ihre Verbindungsanschlüsse zu überwachen. Dies ist der Grund, warum zu einem späteren Zeitpunkt kein Benutzerprozess mehr die Möglichkeit hat, eine Verbindung zu einem dieser Anschlüsse herzustellen. Die Verbindungsanforderung würde nie beantwortet werden.

2.4.2 Umsetzungen der C2-Sicherheitsrichtlinien

Diese Kapitel erläutert, wie Windows 2000 die Anforderungen der C2-Sicherheitseinstufung, mit Hilfe der Sicherheitssystemarchitektur, implementiert. Zu erst wird die Realisierung des oben aufgeführten Sicherheitseinstufungskriteriums 4 beschrieben.

Schützen von Objekten

Die Grundlage der Zugriffskontrolle und Überwachung in Windows 2000 bilden der Objektschutz und die Kontoführung über Zugriffe auf Objekte. Zu den schützenswerten

Objekten zählen z. B. Dateien, Geräte, Pipes, Prozesse, Threads, Ereignisse, Arbeitsstationen, Desktops, Dienste, Drucker usw.

Zugriffsüberprüfungen
Alle Systemressourcen sind als Objekte im Kernelmodus implementiert. Zur Nutzung im Benutzermodus müssen diese Objekte exportiert werden. Dieser Export fordert eine Sicherheitsüberprüfung an. Bei der Realisierung der Objektsicherheit spielt der Objekt-Manager eine große Rolle. Zur Steuerung des Zugriffes muss das System zu erst einmal Gewissheit über die Identität des Benutzers haben. Um die Gewissheit zu erhalten verlangt Windows 2000 eine authentifizierte Anmeldung vor dem Zugriff auf Ressourcen. Erwünscht ein Prozess ein Handle auf ein Objekt, muss der Objekt-Manager und das Sicherheitssystem anhand der Sicherheitskennung des Aufrufers entscheiden, ob dem Aufrufer ein Handle zugewiesen werden soll, der dem Prozess Zugriff auf das gewünschte Objekt gewährt. Wird dem Prozess Zugriff auf das Objekt gegeben, hat automatisch jeder Thread des Prozesses Zugriff, da alle die gleiche Handletabelle benutzen. Die Zugriffsüberprüfung wird über von Windows 2000 bereitgestellte Funktionen, wie ObpCreateHandle, ExCreateHandle, ObChekObjectAccess und ObpIncrementHandleCount realisiert. Je nach dem auf welches Objekt zugegriffen werden soll, werden andere Funktionen verwendet.

Sicherheitskennungen
Das Sicherheitsmodell des SRM basiert auf einer Gleichung mit drei Eingabeparametern; die Sicherheitsidentität des Threads; die Zugriffsart, die der Thread anfordert und die Sicherheitseinstellungen des Objekts. Das Ergebnis einer Anfrage ist entweder "ja"(Zugriff gewährt) oder "nein"(Zugriff nicht gewährt). Windows 2000 verwendet Sicherheitskennungen (Security Identifiers, SIDs) zur Identifikation von Entitäten (führen Operationen aus). Benutzer, sowie lokale Gruppen, Domänengruppen, lokale Computer, Domänen und Domänenmitglieder haben SIDs. Solch eine SID ist ein numerischer Wert variabler Länge, der aus einer Versionsnummer, der SID-Struktur, einem 48 Bit langen Wert für die ausstellende Autorität und einer variablen Anzahl von 32-Bit-Werten für die untergeordnete Autorität (relative Kennung, auch RID für relative Identifier genannt) besteht. Der Wert für die ausstellende Autorität bezeichnet den Agenten, der die SID ausgestellt hat, wobei es sich in der Regel um ein lokales Windows 2000-System oder eine Domäne handelt. Der Wert für die untergeordnete Autorität identifiziert Vertrauensnehmer der ausstellenden Autorität und RIDs stellen für Windows 2000 eine Möglichkeit dar, auf der Grundlage einer SID eindeutige SIDs zu generieren. Da SIDs lang sind und Windows 2000 sicherstellt, dass innerhalb jeder SID echte Zufallswerte generiert werden, ist es praktisch unmöglich, dass Windows 2000 auf Rechnern oder Domänen irgendwo zwei identische SIDs erstellt. Um sich nun als ein anderer auszugeben, müsste man die SID verändern. Eine Beispiel-SID in Textform ist S-1-5-21-1463437245-1224812800-863842198-1128. Jeder SID steht das Präfix S voran. Die dargestellte SID hat die Versionsnummer 1, den Wert 5 für die ausstellende Autorität, sowie vier Werte für untergeordnete Autoritäten plus eine RID (1128). Hierbei handelt es sich um eine Domänen-SID. Ein lokaler Computer dieser Domäne würde eine SID mit derselben Versionsnummer, dem

gleichen Autoritätswert und der gleichen Anzahl von Werten für untergeordnete Autoritäten haben.

Token
Der Token ist ein Objekt mit dessen Hilfe der SRM den Sicherheitskontext eines Prozesses oder eines Threads identifiziert. Der Sicherheitskontext besteht aus Informationen, die die Berechtigungen, Konten und Gruppen beschreiben, die dem Prozess bzw. Thread zugeordnet sind. Zu Beginn einer Sitzung erstellt Winlogon einen Starttoken, der den sich anmeldenden Benutzer repräsentiert. Alle vom Benutzer ausgeführten Programme erben eine Kopie des Starttokens. Trotz unterschiedlicher Größen von Token, die unterschiedliche Benutzer besitzen, enthalten alle die in Abbildung 2 dargestellten Daten.

Abbildung 2.2: Zugriffskonten

Sicherheitsbeschreibungen und Zugriffssteuerung

Die eben behandelten Token sind nur ein Teil der Objektsicherheitsgleichung. Auf der anderen Seite stehen die Sicherheitsinformationen, die mit einem Objekt verknüpft sind. Diese geben an, welche Operationen mit diesem Objekt ausgeführt werden können. Sie implementieren zugleich das 2. Kriterium der C2-Einstufungen. Die zur Speicherung vorgesehene Datenstruktur wird Sicherheitsbeschreibung genannt und umfasst folgende Attribute:

Versionsnummer: Version des SRM-Sicherheitsmodells, mit dem die Beschreibung erstellt wurde.

Flags: Optionale Bezeichner, die das Verhalten oder die Eigenschaft einer Beschreibung definieren.

Besitzer-SID: Sicherheitskennung des Besitzers.

Gruppen-SID: Sicherheitskennung der Primärgruppe des Objekts.

Freigegebene Zugriffskontrolllist (DACL): Legt fest, wer welchen Zugriff auf ein Objekt erhält.

Systemzugriffskontrollliste (SACL): Legt fest, welche Operationen von welchen Benutzern im Protokoll der Sicherheitsüberwachung verzeichnet werden soll.

Eine Zugriffskontrollliste (ACL für Access Control List) besteht aus einem Header und null oder mehr Zugriffskontrolleintrags-Strukturen (ACE - Access Control Entry). Es gibt, wie oben schon aufgeführt, zwei Typen, die DACLs und SACLs. In einer DACL enthält jeder ACE eine SID und eine Zugriffsmaske (Reihe von Flags). Es können genau vier ACEs vorkommen; access-allowed (Zugriff erlaub), access-denied (Zugriff verweigert), allowed-object (Objekt erlaub) und denied-object (Objekt verweigert). Die ACEs bestimmen die einzelnen Rechte der Nutzer auf dem Objekt. Eine SACL enthält zwei Typen von ACEs, die Systemüberwachungs-ACEs und Systemüberwachungsobjet-ACEs. Sie legen fest, welche Operationen, die von Benutzern oder Gruppen auf einem Objekt ausgeführt werden, überwacht werden sollen. Die Ergebnisse der Überwachung werden in dem Überwachungsprotokoll des Systems gespeichert.

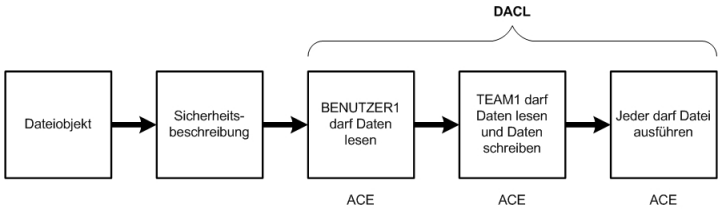


Abbildung 2.3: Freigegebene Zugriffskontrollliste (DACL)

Sicherheitsüberwachung

Mit Hilfe dieser Funktion wird die C2-Einstufungsrichtlinie 3 umgesetzt. Die Überwachungsrichtlinien legen fest, ob ein bestimmter Typ von Sicherheitsrichtlinien überwacht wird. Diese Überwachungsrichtlinien sind Bestandteil der lokalen Systemrichtlinien und werden vom Lsass auf dem lokalen System verwaltet. Lsass sendet während des Systeminitialisierung und sofort bei Änderung der geltenden Überwachungsrichtlinien Nachrichten an den SRM um ihn zu informieren. SRM bearbeitet diese Nachrichten mit Hilfe von Überwachungsereignissen, die aus einem geschützten Teilsystem übergeben bzw. abgefangen werden. Das Ergebnis, die Überwachungs- datensätze werden dann wieder an den Lsass zurücksgesandt. Lsass ist dann verantwortlich für deren Berarbeitung, hinzufügen von einschlägigen Details (z. B. Informationen, die zur Identifizierung des überwachten Prozesses erforderlich sind), und Weiterleitung an den Ereignisprotokollanten (Event Logger) über RPC (Remote Procedure Call). Der Ereignisprotokollant-Thread schreibt den Überwachungsdatensatz daraufhin in das Sicherheitsereignisprotokoll. Die Überwachungsdatensätze werden jeweils in einer Warteschlange gesammelt und in der Reihenfolge ihres Eingangs an den LSA weitergeleitet. Es gibt zwei Möglichkeiten

der Weiterleitung. Diese sind abhängig von der Größe der Überwachungsdatensätze. Kleine Überwachungsdatensätze (kleiner als die maximale LPC-Nachrichtengröße) werden als LPC-Nachricht an den LSA übermittelt. Dabei werden die Überwachungsdatensätze einfach aus dem Adressraum des SRM in den Adressraum des Lsass-Prozess kopiert. Bei großen Datensätzen nutzen der SRM und die LSA einen gemeinsamen Speicherbereich. In der LPC-Nachricht wird in diesem Zusammenhang einfach ein Zeiger auf den Speicherbereich des Überwachungsdatensatzes übergeben. In Abbildung 4 sind die an diesem Prozess beteiligten Komponenten und deren Kommunikation untereinander dargestellt.

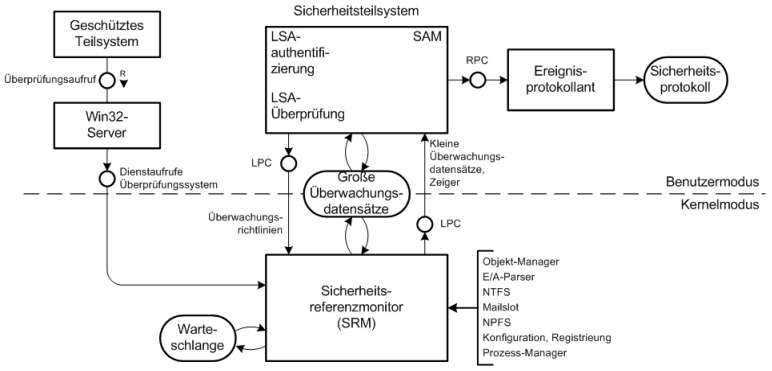


Abbildung 2.4: Weiterleitung der Sicherheitsüberwachungsdatensätze

Sichere Anmeldung

In diesem Abschnitt wird nun beschrieben, wie der 1. Punkt der Anforderungen der C2-Einstufungen umgesetzt wird. Die interaktive Anmeldung involviert den Anmeldeprozess (Winlogon), Lsass, ein oder mehrere Authentifizierungspaket(e) und den SAM oder Active Directory. Authentifizierungspakete sind durch DLL's repräsentiert, die die Authentizitätsprüfungen durchführen. Bei einer interaktiven Anmeldung an einer Domäne verwendet Windows 2000 das Authentifizierungspaket Kerberos.dll und bei einer interaktiven Anmeldung an einem lokalen Computer das Paket Msv1_0.dll. Winlogon ist ein vertrauenswürdiger Prozess, der für die Überwachung und Verwaltung der sicherheitsrelevanten Kommunikation mit dem Benutzer zuständig ist. Neben der Koordination der Anmeldung startet der Anmeldeprozess die Benutzer-Shell während des Anmeldevorgangs, handhabt die Abmeldung und verwaltet verschiedene andere sicherheitsrelevante Operationen, z. B. die Kennworteingabe und das Sperren bzw. Entsperren der Arbeitsstation. Es muss gewährleistet werden, dass diese Operationen für andere Prozesse nicht sichtbar sind und somit die Kontrolle über den Desktop in deren Hände fällt. Damit wird sichergestellt, dass fremde Prozesse keinen Zugriff auf das Kennwort bekommen. Mit Hilfe der Msgina, der Standard-GINA-DLL, zu finden unter \Winnt\System32\Msgina.dll, werden der Kontoname und das Kennwort des Benutzers ermittelt. Msgina zeigt das bekannte Standardanmeldedialogfeld von Windows 2000 an. Windows 2000 sieht vor, dass die Msgina durch andere GINA's ersetzt werden kann. So kann z. B. ein Fremdhersteller eine GINA implementieren,

die Benutzer über Fingerabdrücken auf Geräten identifiziert und deren Kennwörter aus einer verschlüsselten Datenbank einliest und verwendet. Winlogon ist der einzige Prozess, der die Anmeldeinformationen, aus dem Fenster, von der Tastatur abfangen kann. Nachdem der Anmeldeprozess die Anmeldedaten, Benutzername und Passwort, erhalten hat, ruft er den Lsass auf, welcher versucht den anmeldenden Benutzer zu authentifizieren. Dazu führt er die entsprechenden Routinen (Kerberos.dll oder Msv0_1.dll aus) aus, die auf die entsprechenden Datenbanken (Active Directory für die Anmeldung an einer Domäne oder SAM-Server für die Anmeldung am lokalen Rechner) zugreifen um Benutzer und Kennwort zu verifizieren. Ist die Authentifizierung erfolgreich, aktiviert Winlogon die Anmelde-Shell für den zugehörigen Benutzer. Die elementaren Schritte des eben kurz erklärten Ablaufs des Anmeldevorgangs eines Benutzers sind in Abbildung 5 dargestellt.

2.5 Zusammenfassung

Abschließend kann gesagt werden, dass Windows 2000 ein Vielzahl von Sicherheitsfunktionen zur Verfügung stellt, die sowohl die wichtigsten Anforderungen von Registrierungsbehörden als auch kommerziellen Einrichtungen erfüllen. Im ersten Abschnitt, wurde auf die unterschiedlichen Policies von Windows 2000 eingegangen, um in den folgenden Kapiteln einen knappen Überblick über die internen Komponenten dieses Betriebssystems und deren gesamtheitlichen und sicherheitsrelevanten Bedeutung zu verschaffen. Es wurde insbesondere auf die Grundlagen der von Windows 2000 zur Verfügung gestellten und durch C2 geforderten Sicherheitsfunktionen eingegangen. Jetzt bleibt nur noch abzuwarten, ob Windows 2000 mit der C2-Sicherheitsstufe evaluiert wird.

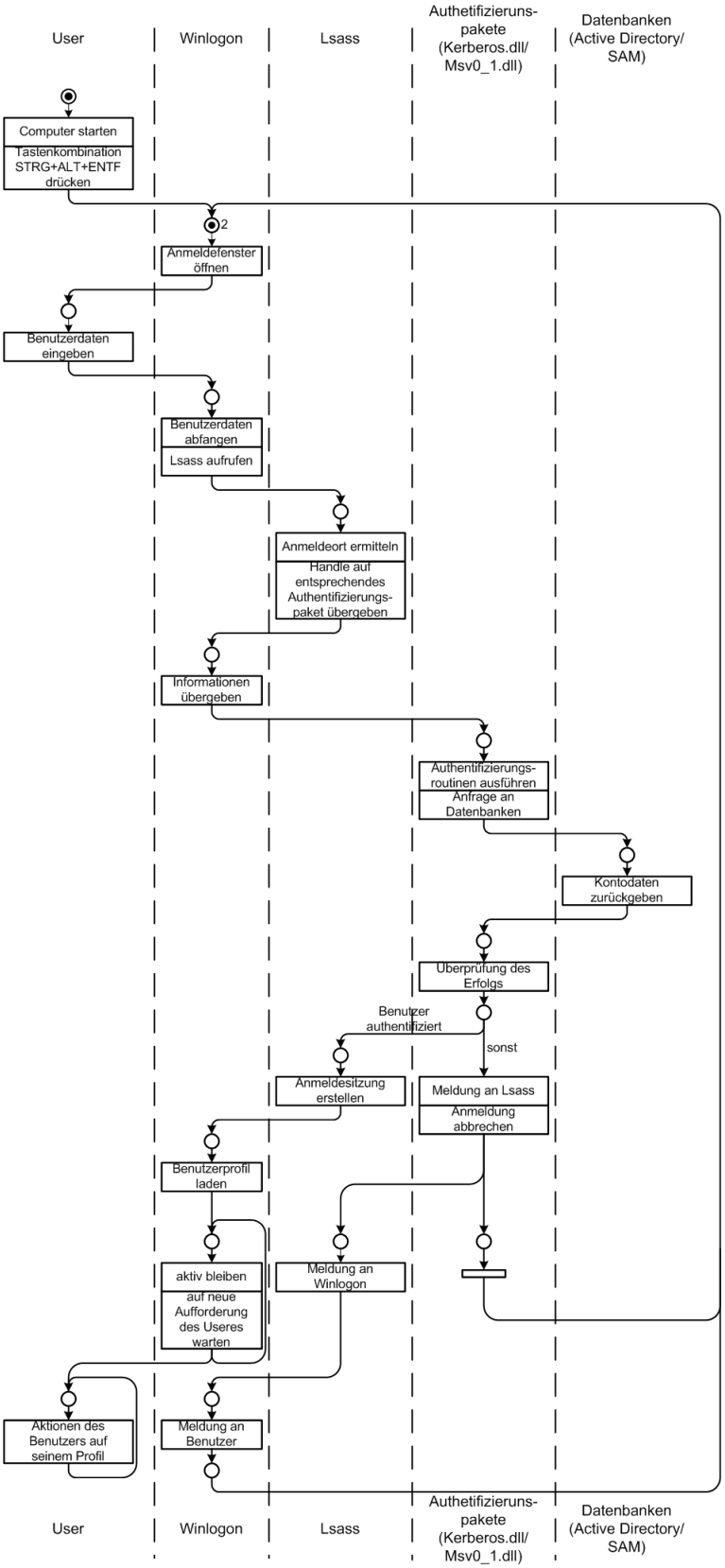


Abbildung 2.5: Ablauf des Anmeldevorgangs eines Benutzers

Literaturverzeichnis

[1] Gregg Branham. Windows NT Domain Architectur. *Macmillan Technical Publishing*, 1, 1999.

[2] Microsoft Corporation. Microsoft Windows NT. *Microsoft Press*, 1, 1993.

[3] Elen Custer. Inside Windows NT. *Microsoft Press*, 1, 1993.

[4] Mark Russinovich David Solomon. Inside Windows 2000. *Microsoft Press*, 3, 2001.

[5] Olaf G. Koch. Windows NT 4 Server. *Markt&Technik*, 1, 1996.

3 DCOM, .Net: Mechanismen der Durchsetzung und Verwaltung von Rechten und Politiken

Dennis Brockhoff, Mathias Fritzsche

Abstract

Immer wieder kommt es zu medienträchtigen Angriffen auf das Windows Betriebssystem.
Dabei wird klar, wie wichtig und aktuell umfassende Sicherheitsmechanismen sind. Besonders durch den stetigen Anstieg verteilter Systeme werden umfassende Sicherheitskonzepte immer wichtiger. In dieser Ausarbeitung werden nun Mechanismen vorgestellt und beurteilt, wie bei DCOM und .Net Sicherheit in verteilten Systemen realisiert wird. Wichtige Punkte sind dabei die unterschiedlichen Herangehensweisen von DCOM und .Net in Bezug auf die Durchsetzung und Verwaltung von Rechten und Politiken.

3.1 Einleitung

Durch Rechte wird festgelegt, welche Aktivitäten und Zugrffe einem Benutzer oder einer Gruppe von Benutzern erlaubt sind ([2]). Nach der MSDN ([2]) stellen Politiken nun gleichmäßig bestimmte Sicherheitsstandards für Benutzer oder Gruppen sicher. Sie bieten demnach eine Basissicherheit einer Entwicklung. Im Gegensatz zu Benutzerrechten gelten Politken für alle Benutzer und Objekte einer Entwicklung.
Diese Ausarbeitung beschäftigt sich mit dem Thema, durch welche Mechanismen bei DCOM und .Net Rechte und Politken in Bezug auf Sicherheit umgesetzt und verwaltet werden können. Außerdem werden die grundsätzlichen Unterschiede zu diesem Thema zwischen dem DCOM und dem .Net Ansatz vorgestellt.
Der 5. Abschnitt geht deshalb auf die Realisierungen von Rechten und Politiken durch DCOM ein und zeigt dabei die Probleme auf, die DCOM nicht löst. Der Abschnitt 6 beschreibt dann wie .Net diese Probleme aufgreift und welche Lösungsansätze von .Net realisiert werden.
Der folgende Abschnitt beschäftigt sich einleitend mit der Definition von Middleware, mit Anforderungen an Sicherheit von Middleware sowie mit Bedrohungen, vor denen Middleware geschützt werden muss. Durch diesen Abschnitt soll besonders die Notwendigkeit von Rechten und Politiken deutlich werden.

3.2 Definition von Sicherhet bei Middleware

Bereits 1997 hat die OMG die vier Anforderungen Vertraulichkeit, Integrität, Verantwortlichkeit und Verfügbarkeit an Sicherheit bei Middleware formuliert ([7]).

Vertraulichkeit bedeutet dabei, dass nur autorisierte Benutzer Zugriff auf bestimmte Informationen erhalten. Integrität verbietet eine Modifikation von Informationen durch nicht autorisierte Benutzer. Verantwortlichkeit heißt, dass Nutzer einer Middleware selbst die Verantwortung für sicherheitsrelevante Information tragen, sich also aktiv an der Sicherung von Informationen beteiligen müssen. Mit Verfügbarkeit ist gemeint, dass ein Sicherheitskonzept sicherstellen muss, dass berechtigten Benutzer Zugriff auf bestimmte Informationen gewährt werden muss.

Diesen Anforderungen an Sicherheit bei Middleware stehen Bedrohungen gegenüber, die 1994 von Harold W. Lockard ([9]) beschrieben wurden. Für ihn gibt es die Bedrohung des Unerlaubten Zugriffs auf Informationen, der Modifikation von Information, des Diebstahls und Missbrauchs von Diensten sowie der Vortäuschung einer falschen Identität.

Um diese Bedrohungen zu umgehen und Sicherheit zu realisieren definiert DCOM und .Net bestimmte Funktionalitäten. Durch diese Funktionalitäten sollen Rechte und Politiken realisierbar sein, die die genannten Bedrohungen verhindern sollen. Die folgenden Abschnitte beschreiben deshalb, welche Funktionalitäten bei DCOM und .Net verfügbar sind um Sicherheit zu realiseren. Auch der Aspekt der Administration von Sicherheit wird beschrieben.
Dabei ist der Genauigkeit wegen zu beachten, dass es sich bei dem .Net Framework nicht nur um ein Framework zur Realisierung von Middleware handelt, wie später genauer beschrieben wird.

3.3 Umsetzung von Sicherheit bei DCOM

In diesem Abschnitt wird ein Überblick über DCOM gegeben. Im Hauptteil kommt es zu einer Beschreibung von Aspekten und Realisierungen von DCOM sowie von Nachteilen und Sicherheitslücken.

3.3.1 Übersicht über DCOM

Nach [2] ist „DCOM (Distributed COM) eine Technologie zur Kopplung von Anwendungen auf heterogenen Systemen.“ Ziel von DCOM ist es Programme, also Dienste über ein Netzwerk zur Verfügung zu stellen.
Die Grundlage von DCOM ist COM. Dies definiert, wie Komponenten, die Dienste zur Verfügung stellen, durch Clients angesprochen werden können.
Diese Client Server Kommunikation kommt bei COM aber nicht ohne Zwischenakteure aus. Dies liegt daran, dass Prozesse in Betriebssystemen voneinander gekapselt sind.
Deswegen kann ein Client eine Komponente nicht direkt aufrufen, sondern muss

eine Form der Interprozesskommunikation, und somit Funktionalität des Betriebssystems nutzen. COM stellt nun genau diese Funktionalität zur Interprozesskommunikation zur Verfügung, wie die folgende Abbildung zeigt. Auf die einzelnen Komponenten der Interprozesskommunikation wird in der MSDN ([2]) eingegangen. Die wichtigsten Akteure werden aber auch im Folgenden beschrieben.

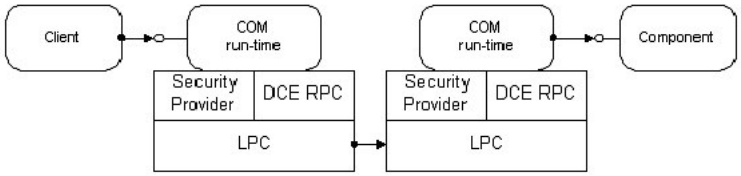


Abbildung 3.1: COM Übersicht ([2])

DCOM ist nun eine Erweiterung von COM in dem Sinne, dass die lokale Interprozesskommunikation durch ein Netzwerkprotokoll ersetzt wird und somit eine Kommunikation zwischen einem Client und einem Server über Rechnergrenzen hinweg möglich wird. Auch hier ist wieder die Realisierung der eigentlichen Kommunikation vor den Client und den Serveranwendungen verborgen.
Die folgende Abbildung zeigt nun diese erweiterte COM Architektur. Man sieht, dass der LPC, also der Local Procedure Call von Abbildung 1 bei Abbildung 2 durch einen Protocol Stack abgelöst wird. Realisiert wird dies durch die Möglichkeit von COM sogenannte Loadable Modules einzubinden. Bei DCOM werden Netzwerkprotokolle durch solche Module eingebunden.

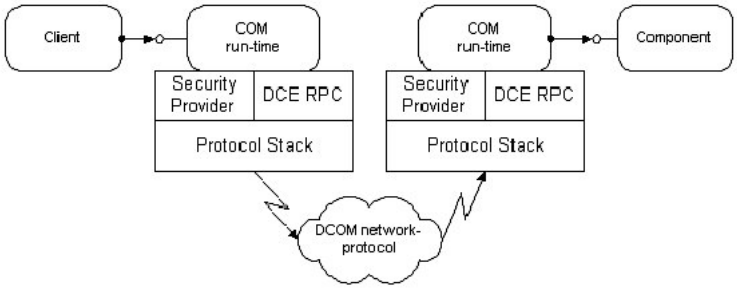


Abbildung 3.2: DCOM RPC ([2])

Als Komponente der Abbildungen 1 und 2 ist ein Security Provider abgebildet. Die Funktionalität dieser DCOM Komponente wird im folgenden Abschnitt beschrieben.

3.3.2 Aspekte von DCOM Sicherheit

Im Abschnitt 3 wurden die Bedrohungen des unerlaubten Zugriffs auf Information, der Modifikation von Information, des Diebstahls und Missbrauchs von Diensten

sowie der Vortäuschung einer falschen Identität eingeführt.
In diesem Abschnitt wird nun untersucht, wie der DCOM Security Provider vor diesen Bedrohungen schützt. Abschließend wird untersucht werden, welche Nachteile und Probleme sich bei dieser Realisierung ergeben.

Das Sicherheitskonzept von DCOM umfasst die beiden wichtigen Konzepte Authentification und Access Control.
Dabei muss allerdings beachtet werden, dass alle Security Konzepte von DCOM durch das entsprechende Windows Betriebssystem bereitgestellt werden. Da es Unterschiede zwischen den Realisierungen bei dem Microsoft Windows NT und dem Microsoft Windows 2000 Betriebssystem gibt ergeben sich auch Inkompatibilitäten zwischen Sicherheitseinstellungen von Komponenten unterschiedlicher Windows Versionen. Beispielsweise wird dies bei der Tatsache deutlich, dass Windows NT Versionen vor NT 4.0 keine Unterstützung für COM Security bietet. Somit konnte bei diesem „Sicherheitkonzept“ ein beliebiges Objekt ein beliebiges anderes Objekt instanziieren.

Authentifizierung

Authentifizierung hat das Ziel die Identifizierung eines Nutzers zu realisieren und eine Vortäuschung einer falschen Identität zu vermeiden. Bei DCOM ist es nun möglich ein sogenanntes Authentification Level zu definieren. Dieses gibt an, welche Algorithmen ein Client oder ein Server von dem SSP (Security Service Provider) verlangt, damit sich andere Objekt bei ihm anmelden können.
Im Folgenden werden die Authentification Level beschrieben. Dabei wird erläutert, dass Authentification Level von DCOM nicht nur reine Mechanismen zur Authentifizierung bereitstellen, sondern auch zur Verschlüsselung von Paketen.
Das einfachste Authentification Level ist „None“ und wird durch das Makro RPC_C_AUTHN_LEVEL_NONE definiert. Bei diesem Level wird keinerlei Authentifizierung während der Kommunikation zwischen Client und Server verlangt und es werden prinzipiell alle anderen Sicherheitseinstellungen ignoriert. Es ist leicht einzusehen, dass sich durch diesen Punkt Abhängigkeiten ergeben können, die zu langwierigen Fehlersuchen führen können.
Das nächst stärkere Level ist „Connect“ (RPC_C_AUTHN_LEVEL_CONNECT). Bei diesem Level kommt es durch einem „Handshake“ Algorithmus zur Authentifizierung zwischen Client und Server. „Handshake“ bedeutet dabei, dass Anmeldeformalitäten zwischen Client uns Derver ausgetauscht werden. Im konkreten Fall des „Connect“ Levels wird ein Session Key ausgetauscht, der aber für die weitere Kommunikation nicht genutzt wird. Somit ist sämtliche Kommunikation nach dem „Handshake“ unsicher.
Ein weiteres Level ist „Call“ (RPC_C_AUTHN_LEVEL_CALL). Dabei wird der Header des ersten Paketes eines Aufrufs signiert und überprüft. Der restliche Datenaustausch wird weder signiert, noch verschlüsselt.
Bei dem nächsten Level („Packet“, RPC_C_AUTHN_LEVEL_PKT) wird der Header jedes Paketes signiert, aber nicht verschlüsselt. Der Rest eines Paketes wird aber weder verschlüsselt nach signiert.

Das Level „Packet Integrity“ (RPC_C_AUTHN_LEVEL_PKT_INTEGRITY) bedeutet, dass der gesamte Datenaustausch zwischen Client und Server vom Sender signiert, aber nicht verschlüsselt wird. Der Empfänger hat somit für den gesamten Verkehr den Beweis, dass er von einem bestimmten Sender gesendet wurde.
Das höchste Level ist „Packet Privacy“ (RPC_C_AUTHN_LEVEL_PKT_PRIVACY) und bedeutet, dass jedes Paket zwischen Sender und Empfänger sowohl signiert, als auch verschlüsselt wird.

Die Festsetzung des Authentification Levels kann einerseits bei der Entwicklung der entsprechenden Komponente als Parameter von Funktion CoInitializeSecurity hart codiert werden, andererseits in der Registry für eine Komponente wie abgebildet festgelegt und werden. Die folgenden beiden Abbildungen 3 und 4 zeigen, wie dieser Wert durch Änderung der Registry verwaltet wird.

HKEY_LOCAL_MACHINE\Software\Classes\AppID
{*AppID_GUID*}\AuthenticationLevel = <*authentication_level*>

Abbildung 3.3: Konfiguration des Security Levels als Parameter durch direkte Editierung der Registry

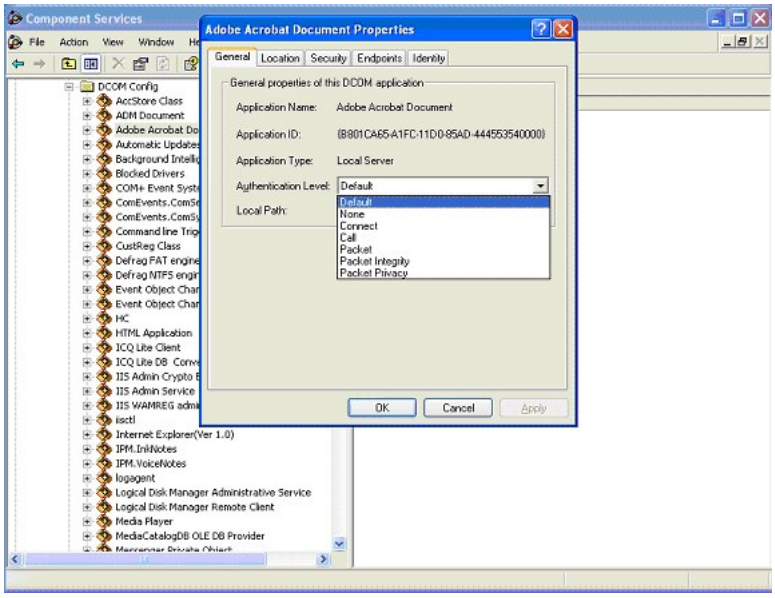


Abbildung 3.4: Configuration des Security Levels durch DCOMCNFG

Zur Realisierung der Anforderungen der unterschiedlichen Security Level existieren Pakete, wie z.B. das Paket RPC_C_AUTHN_GSS_KERBEROS, das eine Kerberos Verschlüsselung bereitstellt.
Zusammenfassend stellt man fest, dass DCOM ab NT Version 4.0 umfassende Mechanismen zur Authentifizierung bietet. Bei der Konfiguration muss allerdings

beachten werden, dass diese Einfluss auf die Access Permissions haben, die im folgenden Abschnitt beschrieben werden.

Access Control

Bei der Access Control geht es um die Ermittlung von Attributen und Privilegien eines Benutzers oder eines Dienstes. Dabei werden 2 Typen von Permissions unterschieden. Einerseits existieren Launch Permissions, die festlegen, wer einen Serverprozess starten darf. Diese lassen sich in der Registry konfigurieren.
Des Weiteren existieren Access Permissions die Objekt-, Methoden- oder Parameterbasiert festlegen, wer auf einen Server zugreifen darf. Access Permissions werden durch so genannte SECURITY_DESCRIPTORS als Bestandteil der DACL (Descretionary Access Control List) beschrieben. Bei jedem Aufruf eines Dienstes kann nun eine Access Check auf Basis der ACL gemacht werden. Wenn so ein Check fehlschlägt wird keine Client Server Verbindung zustande kommen.
Dabei ist allerdings zu beachten, dass Access Control Lists nur von solchen Anwendungen aufgerufen werden dürfen, die die Funktion CoInitializeSecurity nicht benutzen. Es handelt sich somit um ein Konzept, das parallel zu dem bereits beschriebenen im letzten Abschnitt existiert. Die folgende Abbildung zeigt eine beispielhafte Konfiguration einer Access Permission in der Registry.

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole
DefaultAccessPermission = *ACL*

Abbildung 3.5: Konfiguration einer Access Permission durch direkte Editierung der Registry

3.4 Beurteilung von DCOM Sicherheit

Im Abschnitt 2 wurden die Bedrohungen des unerlaubten Zugriffs auf Informationen, der Modifikation von Information, des Diebstahls und Missbrauchs von Diensten sowie der Vortäuschung einer falschen Identität vorgestellt.
DCOM stellt nun Mechanismen bereit, um Rechte und Politken umzusetzen und somit Vorkehrungen gegen jede dieser Bedrohungen zu treffen. So kann beispielsweise das Vortäuschen einer falschen Identität durch entsprechenden Authentifizierungslevel, wie sie im vorangegangenen Abschnitt beschrieben sind, verhindert werden.
Zusätzlich gibt es mehrere Konzepte, wie bei COM/DCOM Sicherheit realisiert werden kann. Das Konzept der Authentification und der Access Control wurden im letzten Abschnitt erläutert. Außerdem wurde gezeigt, dass man Sicherheit einerseits durch Konfiguration an der Registry, andererseits durch hartes Codieren konfigurieren bzw. verwalten kann.

Das Sicherheitskonzept von COM/DCOM hat allerdings auch einige Nachteile. Zum einen kann die Vielzahl an Möglichkeiten der Implementierung und Adminis-

tration von Sicherheitsaspekten zu einem sehr hohen Implementations- und Administationsaufwand ausarten. Dies wird vor allem dadurch verstärkt, dass einige in Konfigurationen Abhängigkeit mit anderen stehen, wie z. B. bei dem Authentification Level „None“ im letzten Abschnitt gezeigt wurde. Zudem ist es oftmals nicht zweckmäßig jedes COM oder DCOM Objekt einzeln in der Registry zu administrieren. Ein hoher Administrationsaufwand bewirkt sehr oft, dass auf Sicherheitsapekte, die eigentlich möglich sind, verzichtet wird.

Zusätzlich sind die Möglichkeiten der Konfiguration von Sicherheit einer DCOM Anwendung starr und eingeschränkt. Beispielsweise kann nur das Authentifizierungslevel administriert werden, aber nicht die spezielle Art der Authentifizierung (z.B. URL, ID, Name). Außerdem gibt es kaum Möglichkeiten eines Objektes flexibel Beweise (z.B. URL, Key) zu erbringen, um ganz bestimmte Zugriffe zu erhalten, die unabhängig von den Nutzerrechten sind. Somit lassen sich keine Szenarien realisieren, bei denen Objekte nicht automatisch die gleichen Rechte wie der entsprechende Nutzer haben. Im Falle eines Administrators kann das verherende Auswirkungen haben, sobald er fremden Code zu Ausführung bringt.

Außerdem immer wieder Sicherheitslücken von DCOM auf, wie z.B. der Blaster Worm, der durch ein überlaufen eines Protokollstacks Code auf beliebigen Rechnern zum laufen bringen kann.

Das Hauptproblem von DCOM im Allgemeinen ist allerdings die Tatsache, dass es vielen aktuellen Anwendungen durch die Verbundenheit an eine bestimmte Windows Version nicht mehr gerecht wird indem es kaum interoperabel ist. Neue Konzepte, wie z.B. der Ansatz des Ubiquitous Computing (allgegenwärtiges Computing) fordern aber eine starke Interoperabilität, die durch das RPC (Remote Procedure Call) Konzept von DCOM nicht realisiert ist. In Bezug auf Sicherheit kann es schon Probleme geben, wenn eine Komponente, die auf einem Windows NT 4.0 Rechner läuft und eine weitere Komponente, die auf einem Windows 2000 Rechner agiert, kommunizieren wollen. Der Grund dafür liegt darin, dass Windows 2000 Kerberos Verschlüsselung unterstützt, Windows NT 4.0 nicht.

Wünschenswert wäre ein einfach administrierbares und flexibles Sicherheitskonzept das beispielsweise zwischen einer Windows CE und einer Linux Anwendung funktioniert. Dieses Konzept greift .NET begrenzt auf, indem die RPC‘s mit Hilfe von Protokollen realisiert werden, die diese Interoperabilität sicherstellen können. Außerdem gibt es die Unterscheidung zwischen Codesicherheit und Rollenbasierter Sicherheit, was eine stärkere Flexibilität von Sicherheitskonzepten ermöglicht.
Die folgenden Abschnitte beschäftigen sich deshalb mit dem Themen .NET Sicherheit.

3.5 NET

Die folgenden Abschnitte befassen sich mit der Sicherheit in .NET. Dabei werden zuerst Veränderungen im Gebrauch mit Computern behandelt. Anschließend geht es um die Themen rollenbasierter Sicherheit und Codesicherheit. Dabei handelt

es sich im Mechanismen des .NET Frameworks welche Rechte für Nutzer und Zugriffsrechte für Code festlegen.

3.5.1 Probleme

Im Bereich der Hardware geht nach [8] die Entwicklung weg vom Desktop hin zu Client-Server-Anwendungen. Außerdem gibt es immer mehr mobile Endgeräte wie PDA's oder Handys. DCOM ist für ein solches Anwendungsszenario nicht ausgelegt. Denkt man allein an Interoperabilitätsprobleme zwischen Windows NT 3.5 und 4.0 in bezug auf Sicherheit, so kann man sich leicht vorstellen, dass DCOM nicht wirklich für den mobilen Einsatz ausgelegt ist. Weitere Aspekte wurden im letzten Abschnitt erläutert.

3.5.2 Lösungen

NET soll z.B. die Probleme der Interoperabilität zwischen heterogenen Systemen lösen. .NET ist allerdings der Überbegriff über Produkte, Strategien und Technologien, von denen einige im Folgenden beschrieben sind.

- Das .NET Framework ist eine Plattform, auf die im Folgenden genauer eingegangen wird.
- Visual Studio .NET, eine neue Entwicklungsumgebung, die mehrere .NET-Programmiersprachen (z.B. die eigens für .NET geschaffene Sprache C# oder VB.NET) unterstützt
- .NET My Services ist eine Gruppe von Diensten, die Funktionen wie Authentifizierung übernehmen
- NET Enterprise Server, die abgesehen vom Namen unabhängig von den anderen Technologien sind und u.a. die Produkte Exchange Server 2000, Application Center 2000, SQL Server 2000 beinhalten.
- .NET Devices, die durch eine abgespeckte Version des .NET Framework unterstützt werden (.NET Compact Framework).

Im Folgenden wird der Begriff .NET als die Möglichkeiten des .NET Frameworks benutzt. Dabei stehen vor allem Sicherheitsaspekte im Fordergrund.

3.5.3 Sicherheit im .Net Framework

Das .NET-Framework bietet zwei komplementären Formen von Sicherheit an, Benutzersicherheit und Codesicherheit. Mit diesen beiden Mechanismen soll es möglich sein sicheren Code und sichere Webanwendungen zu erstellen. Im Folgenden werden beide vorgestellt. Dafür ist es allerdings notwendig das Prinzip des verwalteten Codes (managed code) von .Net zu beschreiben.

Vorzüge von verwaltetem Code in Bezug auf Sicherheit

.NET Framework-Assemblys werden mit verwaltetem Code erstellt. Compiler für Programmiersprachen, wie das Entwicklungsprogramm Microsoft Visual C# und das .NET-Entwicklungssystem Microsoft Visual Basic, erzeugen MSIL-Anweisungen (Microsoft Intermediate Language), die in den standardmäßigen PE-Dateien (Portable Executable) von Microsoft Windows im Format .dll oder .exe enthalten sind. Durch Verwenden einer Zwischensprache in Verbindung mit der Laufzeitumgebung, die die CLR (Content Language Runtime) zur Verfügung stellt, können Assemblyentwickler unmittelbare Sicherheitsvorteile nutzen. Im Folgenden werden einige beschrieben.

- Überprüfung von Dateiformat und Metadaten. Die CLR überprüft die PE-Datei auf Gültigkeit des Formats und darauf, dass keine Adressen aus der PE-Datei heraus verweisen. Dadurch lässt sich ein Assembly leichter isolieren. Die CLR überprüft auch die Integrität der Metadaten, die in der Assembly enthalten sind.

- Codeüberprüfung. Der MSIL-Code wird auf Typensicherheit und JIT-Kompilierzeit (Just-In-Time Kompilierzeit) geprüft. Dadurch wird die Anfälligkeit für Pufferüberläufe in verwaltetem Code praktisch eliminiert. Allerdings muss dennoch sorgfältig jeder Code, der nicht verwaltete APIs (Application Programming Interfaces) aufruft, auf möglichen Pufferüberlauf geprüft werden.

- Integritätsprüfung. Die Integrität von Assemblys mit starkem Namen wird mit einer digitalen Signatur überprüft. So wird gewährleistet, dass die Assembly seit ihrer Erstellung und Signatur in keiner Weise geändert wurde. Das bedeutet, dass Angreifer den Code des Users nicht durch direkte Manipulation der MSIL-Anweisungen ändern können.

- Codezugriffssicherheit. Aufgrund der virtuellen Ausführungsumgebung, CLR, können zusätzliche Sicherheitsüberprüfungen während der Laufzeit ausgeführt werden. Insbesondere können durch Codezugriffssicherheit mehrere Sicherheitsentscheidungen während der Laufzeit, abhängig von der Identität des aufrufenden Codes, getroffen werden ([5]).

Nachdem nun einige Sicherheitsrelevanten Vorteile von verwalteten Code bei .Net vorgestellt wurden geht es im folgenden Abschnitt um die wesentlichen Aspekte der Benutzersicherheit und der Codesicherheit.

Benutzersicherheit im Vergleich zu Codesicherheit

Benutzersicherheit und Codesicherheit sind zwei sich ergänzende Sicherheitsformen. Benutzersicherheit überprüft die Identität (Authentifikation) und Zugriffsrechte (Authorisation) eines Benutzers. Codesicherheit überprüft woher der Code stammt, wer ihn geschrieben hat und was der Code ausführen darf, dabei wird der Zugriff der Anwendung (nicht des Benutzers) auf Systemressourcen wie z.B. Dateisystem, Registrierungsdatei, Netzwerk, Verzeichnisdienste und Datenbanken

organisiert. Ausschlaggebend ist, wozu der Code berechtigt ist und nicht auf welchem Benutzerkonto er ausgeführt wird.
Die .NET Framework-Benutzersicherheitsimplementierung heißt rollenbasierte Sicherheit. Die Codesicherheitsimplementierung heißt Codezugriffssicherheit.

Codezugriffssicherheit

Codezugriffssicherheit autorisiert Code beim Versuch, auf geschützte Ressourcen wie Dateisystem, Registrierungsdatei und Netzwerk zuzugreifen, oder beim Versuch, andere privilegierte Vorgänge auszuführen, zum Beispiel verwalteten Code aufzurufen.
Codezugriffssicherheit ist ein wichtiger zusätzlicher Verteidigungsmechanismus, der zum Einschränken von Codeelementen verwendet werden kann, denn es kann überprüft werden wer den Code erstellt hat. Wurde er von einem Angreifer erstellt, dessen Identität man nicht kennt und folglich auch nicht „traut“, so hat der Code keinen Zugriff auf geschützte Ressourcen. Ein Administrator kann eine Richtlinie für die Codezugriffssicherheit konfigurieren, um einzuschränken, auf welche Ressourcentypen der Code zugreifen und welche anderen privilegierten Vorgänge er ausführen kann. Die zusätzlichen Einschränkungen der Codezugriffssicherheit können das Ausmaß des Schadens für eine Webanwendung im Falle eines gefährdeten Prozesses begrenzen, wenn ein Angreifer die Kontrolle über einen Webanwendungsprozess übernimmt oder Code einfügt, der innerhalb des Prozesses ausgeführt wird.
Abbildung 6 ([2]) zeigt, wie die Codezugriffssicherheit in einer Webanwendung verwendet wird, um den Zugriff der Anwendung auf Systemressourcen, Ressourcen von anderen Anwendungen und privilegierte Vorgänge, zum Beispiel Aufrufen von nicht verwaltetem Code, einzuschränken.

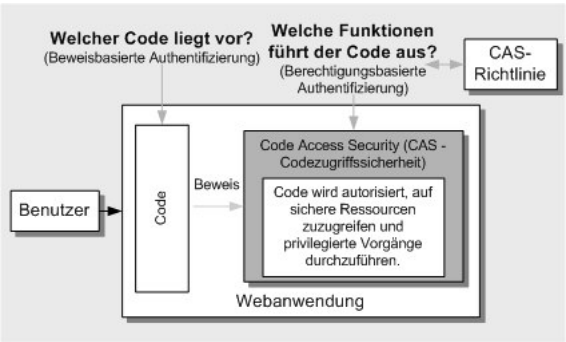


Abbildung 3.6: Darstellung von codebasierter Sicherheit

Die Authentifizierung (Identifikation) von Code basiert auf eindeutigen Codemerkmalen, zum Beispiel auf dessen starken Namen, Herausgeber oder Installationsverzeichnis. Die Autorisierung basiert auf den Codezugriffsgenehmigungen, die in der Sicherheitsrichtlinie für den Code gewährt werden.

Rollenbasierte Sicherheit

Mit Hilfe der rollenbasierten Sicherheit kann eine Webanwendung Sicherheitsentscheidungen abhängig von der Identität oder Rollenmitgliedschaft des Benutzers treffen, der mit der Anwendung interagiert. Wenn die Anwendung die Windows-Authentifizierung verwendet, ist eine Rolle eine Windows-Gruppe. Wenn die Anwendung andere Formen der Authentifizierung verwendet, wird die Rolle von der Anwendung festgelegt, und die Einzelheiten für Benutzer und Rolle werden in der Regel in SQL Server verwaltet oder vom Benutzer gespeichert.
Die Identität des authentifizierten Benutzers und der zugewiesenen Rollenmitgliedschaft wird Webanwendungen durch Principal-Objekte zur Verfügung gestellt, die der aktuellen Webanforderung angehängt sind.
Abbildung 7 ([2]) zeigt, wie die Benutzersicherheit in einer Webanwendung üblicherweise verwendet wird, um den Benutzerzugriff auf Webseiten zu begrenzen.

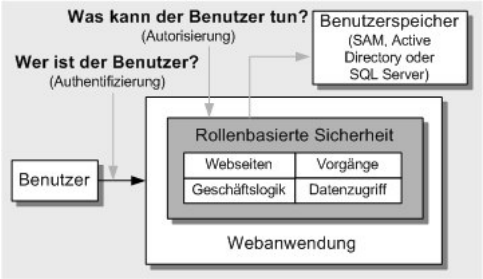


Abbildung 3.7: Darstellung von rollenbasierter Sicherheit

Rollen werden oft genutzt, um bestimmte Regeln umzsetzen. Ein beispielhaftes Szenario ist, dass bei einer Finanzanwendung Überweisungen ab einem bestimmten Wert nur von bestimmten Personen durchgeführt werden können.

Rollenbasierte Sicherheit besteht aus den folgenden Elementen:

- Prinzipale und Identitäten
- PrincipalPermission-Objekte
- Rollenbasierte Sicherheitsüberprüfung
- URL-Autorisierung

Prinzipale und Identitäten: Rollenbasierte Sicherheit wird mit Principal und Identity-Objekten implementiert. Principal-Objekt drücken die Identität und Rollenmitgliedschaft eines authentifizierten Benutzers aus.
Principal-Objekte enthalten auch Identity-Objekte. Zusätzlich zum Benutzernamen sind auch Flags enthalten, die zeigen, ob der Benutzer authentifiziert wurde oder nicht.

PrincipalPermission-Objekte: Das PrincipalPermission-Objekt stellt die Identität und die Rolle dar, über die der aktuelle Prinzipal verfügen muss, um den Code auszuführen. PrincipalPermission-Objekte können deklarativ oder imperativ im Code verwendet werden.
Das deklarativ formulierte Beispiel zeigt, wie nur die Mitglieder der Gruppe Managers Zugriff haben. Das Format des Rollennamens entspricht dem Format MachineName/RoleName oder DomainName/RoleName. Deklarativ veschriebene Sicherheit funktioniert über sogennante Attribute (z.B. PrincipalPermissionAttribute) und hat beispielsweise den Vorteil, dass sie über eine XML Datei konfiguriert werden kann. Im nächsten Abschnitt wird näher darauf eingagangen.

```
[PrincipalPermissionAttribute(SecurityAction.Demand,
Role=@"DOMAINNAME/Managers")]
public sealed class OnlyManagersCanCallMe
{
}
```

Die Klasse OnlyManagersCanCallMe kann also, wie der Name schon sagt, nur von Mitgliedern der Gruppe Manager genutzt werden.

Eine weitere Möglichkeit ist die imperative Formulierung von Sicherheit. Folgendes einfaches Beispiel skizziert deren Verwendung.

```
new PrincipalPermission(null, @"DomainName/WindowsGroup")).Demand();
```

Der Code erstellt ein PrincipalPermission-Objekt, der Benutzername bleibt leer, der Rollenname wird angegeben. Das Codebeispiel zeigt, dass der Benutzername leer geblieben ist, allerdings wird der Rollenname angegeben. Dadurch wird die Identität eines Users überprüft und es werden die Zugriffsrechte der Gruppe verwendet der er angehört. Auf diese Weise können neue Nutzer hinzukommen und die Rechte einer Gruppe übernehmen, ohne dass ein Administrator speziell für ihn Rechte festlegen muss.

Daklarative Sicherheit bietet einerseits viele Vorteile, andererseits können nur mit imperativer Sicherheit Laufzeitvariablen genutzt werden. Diese sind beispielsweise notwendig, wenn ein Nutzer zur Laufzeit einer Gruppe beitritt. Dadurch übernimmt er die Rechte der Gruppe. Aus administrativer Sicht heraus empfielt es sich aber deklarativen Sicherheit zu benutzen. Die folgenden Punkte belegen dies.

- Administratoren oder Assemblybenutzer können genau erkennen und anhand vo XML Dateien konfigurieren, welche Sicherheitsberechtigungen von bestimmten Klassen und Methoden ausgeführt werden müssen.
- Die Leistung wird verbessert. Deklarative Befehle werden nur einmal während des Ladevorgangs ausgewertet, Imperative hingegen wenn die Methode aufgerufen wird.

- Deklarative Überprüfungen auf der Klassenebene werden auf alle Mitglieder der Klasse angewendet. Imperative Überprüfungen werden am Standort des Benutzers angewendet.

Rollenbasierte Sicherheitsüberprüfungen: Für feinstufige Autorisierungsentscheidungen können auch explizite Rollenprüfungen durchgeführt werden, wie die folgende Abbildung verdeutlicht.

```
// Extrahieren des aktuellen Users aus dem HTTP-Kontext
WindowsPrincipal authenticatedUser = User as WindowsPrincipal;

if (null != authenticatedUser)
{

  // Durchführung der Rollenprüfung
  if (authenticatedUser.IsInRole(@"DomainName/Manager") )
  {
    // Benutzer ist berechtigt, Managerfunktionen auszuführen
  }
}
else
{
  // Benutzer ist nicht berechtigt, Managerfunktionen auszuführen
  // Auslösen einer Sicherheitsausnahme
}
```

3.6 Bewertung .NET

Wie gezeigt wurde bietet .NET verschiedene Möglichkeiten um Sicherheit umzusetzen. Grundlegender Ansatz ist dabei das Festlegen von Regeln und Rechten für bestimmte Nutzer bzw. für die Gruppe der sie angehören. Den Zugriff auf geschützte Ressourcen durch fremden Code kann man ebfenfalls begrenzen.
Bei einer Webanwendung hat man immer noch das Problem, dass die Kommunikation über http abläuft. Dieses Protokoll kann man abhören. Es ist also eine Kombination mit SSL erforderlich, um zu gewährleisten, dass die Anwendung sicher wird. Das ist mit der WebRequest-Klasse möglich. Diese Klasse kann mit SSL-fähigen Clients kommunizieren. Die Kommunikation mit Clients die nicht SSL-fähig sind kann von der Webanwendung unterbunden werden.
Insgesamt erscheinen die Sicherheitskonzepte des .NET Frameworks durchdacht und ausgereift, allerdings gibt es auch hier Angriffsmöglichkeiten. In einem golem-Artikel ([6]) wird beschrieben, dass es laut Microsoft eine Sicherheitslücke gibt, wenn ASP.NET-Applikationen im StateServer-Modus laufen, was wiederum das .NET-Framework in der Version 1.0 betrifft. Durch das Sicherheitsloch kann ein Buffer-Overrun in der Cookie-Routine auftreten, der es Angreifern ermöglicht, einen Neustart von

ASP.NET-Programmen auszuführen, was einen Datenverlust nach sich zieht. Unter Umständen verhilft es einem Angreifer auch einen anderen Gruppenstatus zu erlangen, welcher vielleicht den Zugriff auf geschützte Ressourcen erlaubt. Auch können Angreifer so beliebigen Programmcode starten.

3.7 Zusammenfassung

Sowohl das .Net Framework, als auch DCOM implementieren Funktionalität um den Anforderungen und Bedrohungen, die im 3. Abschnitt erläutert wurden zu entsprechen.
Das .NET-Framework bietet dabei allerdings bessere Möglichkeiten um Sicherheit umzusetzen als DCOM. Da .NET die CLR zur Verfügung stellt, können Entwickler von Anwendungen in einer Hochprogrammiersprache (meist C#) Sicherheitsmechanismen implementieren, ohne sich um spezielle Details wie z.B. Speicherverwaltung kümmern zu müssen.
Außerdem basieren beide Systeme auf einem völlig unterschiedlichen Ansatz. DCOM wurde speziell für die Windows-Welt entwickelt und „verträgt“ sich nur schlecht mit anderen Systemen. Aus diesem Grund ist es für die immer mobiler werdenden Welt in der mobile Endgeräte an Bedeutung gewinnen nicht mehr geeignet.
.NET ist durch die Rollensicherheit und die Codesicherheit deutlich darauf ausgelegt in heterogenen Systemen Zugriffsrechte für User und „fremden“ Code festzulegen. Das ist auch zwingend notwendig, da mit der steigenden Zahl mobiler Endgeräte natürlich auch die Zahl potentieller Angreifer steigt. Wenn jeder selbst mit dem Handy beispielsweise per WAP auf Web Services zugreifen kann, sinkt selbstverständlich auch die Hemmschwelle Angriff auf fremde Systeme zu unternehmen, da ein solcher Angriff auf Grund der mobilen Geräte schwere zu seinem Ursprung zurückzuverfolgen ist.
Durch die Rollensicherheit kann man dem entgegenwirken. Man kann Benutzern bzw. Gruppen Rollen zuweisen die nur bestimmte Rechte besitzen.
Die Zukunft wird zeigen ob sich das Konzept von .NET durchsetzen kann und auch kommenden Ansprüchen genügen wird. Bei der momentanen Entwicklung ergibt sich jedoch, dass .NET im Vergleich zu DCOM ein großer Fortschritt im Bereich Sicherheit ist und die derzeitigen Anforderungen deutlich besser erfüllen kann.
Immerhin gestalten sich die Sicherheitskonzepte von .Net um einiges leichter konfigurierbar, indem bei deklarativer Beschreibung von Sicherheitsaspekten lediglich XML Dateien modifiziert werden müssen anstatt Einträge der Registry. Zusätzlich ist der Ansatz von .Net flexibler als der von DCOM. Somit erlaubt .Net besipielsweise Einschränkungen der Rechte von Code, die über die Einschränkungen des entsprechenden Benutzers hinausgehen. Trotzdem gibt es Sicherheitslücken im .Net Framework, von denen einige gezeigt wurden.

Literaturverzeichnis

[1] . http://www.golem.de/0206/20238.html, Juni 2002.

[2] . http://www.galileocomputing.de, April 2004.

[3] . http://www.msdn.microsoft.com, April 2004.

[4] . http://www.delphi-source.de/grundlagen/dotnet/ueberdotnet/framework.php, April 2004.

[5] . http://www.delphi-source.de/grundlagen/dotnet/ueberdotnet/framework.php, Mai 2004.

[6] . http://www.microsoft.com/germany/ms/security/guidance/modules/secmod79.html, April 2004.

[7] . http://www.omg.org/, Mai 2004.

[8] Gerit Kalkbrenner. Mobile Management of local Infrastructure. *Published on IEEE Softcom 2002*, 1:1–6, 2002.

[9] Harold W. Lockard. Guide to developing applications. *OSF DCE*, 1, 1994.

4 Unix: Mechanismen zur Durchsetzung und Verwaltung von Rechten und Politiken

Martin Hoffmann, Steffen Jurrack

Abstract

Dieses Papier befasst sich mit dem Thema Unix Security. Im folgenden werden Sicherheitsmechanismen vorgestellt die Unixsysteme bieten und Hilfestellungen gegeben wie man sein System so konfiguriert, dass es für Angreifer schwerer zugänglich ist. Dabei wird genauer auf die Bereiche Account-Security, Dateisystem-Security und Netzwerk-Security eingegangen. Im weiteren werden noch ein paar aktuelle Angriffe auf Unixsysteme dargestellt um zu zeigen, wie die Sicherheitsmechanismen umgangen werden können.

4.1 Einleitung

Wenn man von Computersicherheit spricht, muss man sich erst einmal verdeutlichen, was diese eigentlich bedeutet: der wichtigste Punkt ist wohl die Integrität der Daten, d.h. Software und Daten dürfen nicht unbemerkt verändert werden. Desweiteren ist die Vertraulichkeit von Daten essentiel, dies bedeutet, dass Dateien nur von autorisierten Benutzern gelesen werden dürfen. Ein weiterer Aspekt ist die sichere Übertragung von Daten in einer Netzwerkumgebung, denn es sollte gewährleistet sein, dass die Übertragung von einem Rechner zu anderen Rechnern, Geräten oder zum Benutzer nicht ausgespäht werden kann. Darüber hinaus sollte das System also die Hard-und Software so funktionieren, wie der Nutzer es erwartet. Da das Betriebssystem die Schnittstelle zwischen Ressourcen(Hardware, Daten) und Akteuren(Nutzer, Prozesse), die Ressourcen nutzen, darstellt, muss es Mechanismen zur Gewährleistung von Computersicherheit besitzen und durchsetzen. Im folgenden wird betrachtet wie Unix-Betriebssysteme dies erreichen.
Unixsysteme gelten im allgemeinen als recht sicher, jedoch müssen bei Unix ein paar Dinge berücksichtigt werden. Ein Unixsystem ist nur sicher, wenn es entsprechend konfiguriert ist und regelmäßig überprüft wird. Man sollte auch darauf achten, dass alle Nutzer des Systems entsprechende Richtlinien einhalten, das beste Passwort und sichereste System nützen nicht viel, wenn die Systemnutzer sich nicht an bestimmte Sicherheitsrichtlinien halten. Deshalb müssen nicht nur vom Betriebsystem sondern auch von allen Netzwerknutzern bestimmte Regeln

eingehalten werden. Unix selbst liefert, aber eine Reihe von Mechanismen, die die Sicherheit auf unterschiedlichen Ebenen des Bertriebsystems realisieren, die in den folgenden Kapieteln näher erläutert werden. Im zweiten Kapitel dieses Papiers wird Accountsicherheit beschrieben, das folgende Kapitel befasst sich dann mit dem Thema Dateisystemsicherheit. Anschließend wird auf den Bereich der Netzwerksicherheit eingegangen. Die darauf folgenden Kapitel befassen sich mit den Themen Network Information Service (NIS), Network File System (NFS) und Remote File Sharing Service (RFS).

4.2 Account-Sicherheit

4.2.1 Passwörter

Das Passwort ist der wichtigste Teil der UNIX Account-Sicherheit. Falls ein Angreifer das Passwort eines Nutzers entdeckt, kann er sich in das System einloggen und mit den Rechten dieses Nutzers arbeiten. Sollte der benutzte Account dem Superuser gehören, hätte der Angreifer Zugriff auf alle Daten und Programme. Um solch fatale Folgen zu vermeiden, ist es dringend notwendig sichere Passwörter zu verwenden. UNIX bietet nun verschiedene Mechanismen zur Erstellung von sicheren Passwörtern und zur sicheren Speicherung dieser Passwörter. Diese Mechanismen werden nun im Folgenden vorgestellt. Das typische UNIX passwd-Programm, welches zur Änderung von Nutzer-Passwörtern dient, legt einige Beschränkung bei der Passwort-Wahl fest. Es verlangt Passwörter, die aus fünf oder mehr Kleinbuchstaben bestehen oder Passwörter aus vier oder mehr Zeichen, wenn Großbuchstaben oder nichtalphabetische Zeichen enthalten sind. Die Einstellungen für die Passwortkriterien lassen bei einigen Unixen in der Datei /etc/login.defs konfigurieren. Dennoch ist es bei älteren Implementierungen möglich auf unsichere Passwörter zu bestehen, dies erfordert nur die dreimalige Eingabe des Passwortes.
Neuere Implementierungen verlangen fünf bzw. sechs Zeichen. Das passwd-Programm allein reicht aber nicht aus um Nutzer-Accounts vor unberechtigten Zugriffen zu schützen, denn die Wahrscheinlichkeit dafür, dass ein Passwort erraten wird, steigt mit seiner Lebenszeit, das heißt dem Zeitraum in dem das Passwort genutzt wird. Das National Computer Security Center hat folgende Gleichung für die Wahrscheinlichkeit dafür das ein Passwort erraten wird bestimmt:

P = L*R/ S
L ist die Lebenszeit des Passworts, R entspricht der Anzahl der Rate-Versuche pro Zeiteinheit und S ist der Passwortraum, d.h. die Anzahl an möglichen Passwörtern, die sich aus den verwendeten Zeichen und der Passwortlänge ergeben, es gilt folgende Relation:

S = AM.

Somit ist leicht ersichtlich, dass bei kurzen Lebenszeiten eines Passworts sowie einem längeren Passwort aus einem umfangreichen Zeichenalphabet sich die Wahrscheinlichkeit dafür, dass jemand das Passwort errät minimieren lässt.

4.2.2 Passwort Generatoren und Password Aging

Die meisten Unix-Implementierungen bieten Mechanismen, die auf die Minimierung der Wahrscheinlichkeit, dass ein Passwort erraten wird abzielen. Ein solcher Mechanismus sind Passwortgeneratoren. Der Vorteil von Passwortgeneratoren ist die Erzeugung von Passwörtern aus umfangreicheren Alphabeten (meist Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen) mit der Besonderheit das diese mittels Zufallsfunktion erzeugten Zeichenketten in keinem Wörterbuch auftauchen und somit nur mittels Brute-Force-Algorithmus erraten werden können. Der Nachteil besteht darin das diese generierten Passwörter schwerer einprägbar sind. Ein

Beispiel für einen Passwortgenerator ist wieder das passwd-Programm mit der Option -a. Ein weiterer Mechanismus, welcher das Erraten von Passwörtern erschwert ist das Passwort Aging, hierbei wird für jedes Passwort eine Lebenszeit festgelegt nach deren Ablauf das Passwort abläuft und geändert werden muss. Natürlich lässt sich auch die Lebenszeit einstellen, dies ist ebenfalls in der Datei /etc/login.defs möglich.

4.2.3 Shadow Passwortdatei

Nun wo die Erzeugung von sicheren Passwörtern geklärt ist, stellt sich die Frage, wie die Passwörter sicher gespeichert werden können. UNIX speichert alle Passwörter verschlüsselt in der Shadow Passwortdatei, welche nur mit Superuser-Rechten gelesen werden kann. Es gibt noch eine passwd file, welche von jedem Nutzer gelesen werden kann und ältere UNIX-Implementierungen speicherten auch in ihr die verschlüsselten Passwörter, doch war es so jedem der einen Account auf dem System besitzt möglich sich diese Datei zu kopieren und zu versuchen die enthaltenen Passwörter zu entschlüsseln, was beim Passwort des Superusers fatale Folgen hätte.

4.2.4 Verschlüsselung mit crypt

Nun stellt sich natürlich die Frage, wie und vor allem wie sicher sind die Passwörter verschlüsselt.
Das Unix-Programm crypt übernimmt die Verschlüsselung und das Vergleichen von Nutzerpasswörtern. Im Folgenden ist beschrieben welche Algorithmen crypt nutzt.
Die Paßwörter werden mit einem Data Encryption Standard (DES) Verfahren symmetrisch verschlüsselt (genauer: eine Reihe von Nullen wird unter der Verwendung von DES mit dem Benutzer-Paßwort verschlüsselt) und auf der Platte abgelegt. Wenn sich jetzt ein Benutzer einloggen will, so gibt er Benutzername und Paßwort an. Das Programm sucht das verschlüsselte Paßwort raus, verschlüsselt das vom Benutzer angegebene Paßwort und vergleicht diese beiden verschlüsselten Paßwörter. Somit wird das echte Paßwort vom System nie aufgedeckt. Die Schwachstelle ist natürlich klar zu sehen: der Benutzer gibt sein Paßwort im Klartext an. Diese Schwäche haben allerdings alle Systeme, daher kann mit einem Keyboard-Sniffer oder trojanischen Pferden die tun als seien sie Login-Programme enormer Schaden angerichtet werden.

4.2.5 Passwort Policies

Alle bisher erwähnten Mechanismen werden sinnlos, wenn sich die Nutzer selbst nicht an bestimmte Richtlinien ihr Passwort betreffend halten. Somit müssen Passwort Policies formuliert werden, welche die Nutzer in irgendeiner Form akzeptieren müssen. Diese Policies müssen es dem Nutzer verbieten sein Passwort an andere weiterzugeben oder sein Passwort in auf Blöcken, Kalendern usw. aufzuschreiben oder gar online zuspeichern. Vor allem muss in der Policies daraufhin gewiesen werden, dass die Nutzer ihr Passwort unbedingt geheim halten müssen.

4.2.6 Mechanismen zur Überwachung von Accounts

Selbst wenn ein Account durch ein sicheres Passwort geschützt ist, gibt es dennoch für einen Angreifer Wege um sich Zugang zu einem Account zu verschaffen. Unix bietet Mechanismen, welche Login-Vorgänge und die Ausführung von Befehlen überwachen und loggen. In der Datei /usr/adm/lastlog werden die Zeiten der letzten Logins sowie das Terminal und der Host von dem aus der Login stattfand vom Login-Prozess aufgezeichnet. Auch die aktuell angemeldeten User werden in der Datei /etc/utmp gespeichert. Zusätzlich sind Informationen über Terminal und Host verfügbar.
Darüber hinaus werden die Informationen zum Login und Logout der einzelnen User in der Datei /usr/adm/wtmp abgelegt. Das Unix System zeichnet wie schon erwähnt die Ausführung von allen Befehlen sowie den Namen des Nutzers, das Terminal, die Ausführungszeit und -dauer auf.
Alle in diesem Abschnitt dargestellten Mechanismen erschweren es einem Angreifer sich über einen gestohlenen Account Zugang zum System zu verschaffen.

4.3 Dateisystem Sicherheit

Die letzte Verteidigung gegen Angreifer sind die Rechte (Permissions), die das Dateisystem zur Verfügung stellt. Diese Permissions ermöglichen es dem Nutzer zu bestimmen, wer die Erlaubnis hat auf Dateien und Verzeichnissen zuzugreifen (lesen, schreiben, auszuführen). In aktuellen Unixsystemen können die Permissionbits für ein Verzeichnis genutzt werden um Nutzer davon abzuhalten, Dateien zu löschen die ihnen nicht gehören.

4.3.1 Datei Permissions

Jede Datei und jedes Verzeichnis hat drei Sätze von Permissions, der erste Satz zeigt die Rechte, die der Besitzer der Datei hat. Der zweite Satz dient dazu die Rechte für die Gruppe mit der die Datei assoziiert wird zu setzen, der dritte Satz die Rechte die sämtliche Nutzer haben. Jeder der drei Sätze enthält drei Permissionbits: read, write und execute.

Modus	Permissions
777	rwxrwxrwx
640	rw-r- - - - -
721	rwx-w - - -x

Zusätzlich gibt es noch einen vierten Satz von Permissions welcher drei Bits enthält:

- set-user-id: Wenn dieses Bit gesetzt ist, wird das Programm mit den Rechten ausgeführt die der Besitzer der Datei hat.
- set-group-id: Dieses Bit hat den gleichen Sinn, wie das set-user-id Bit nur das ein Programm nun mit den Rechten der Gruppe ausgeführt wird.
- sticky: Wenn dieses Bit auf einem Verzeichnis gesetzt ist, dann dürfen Nutzer in diesem Verzeichnis keine Dateien löschen oder umbenennen.

4.3.2 Access Control List (ACL)

Einige Unix-Systeme nutzen Access Control Lists (ACLs), die es ermöglichen für einzelne Nutzer und Gruppen Permissionbits zu setzen.Eine ACL ist dabei eine Liste von (Nutzer, Permission) oder (Nutzer, Gruppe, Permission). Hier ein Beispiel:

(max, *, rwx)
(karl, *, r)
(peter, stud, rw)
(*, stud, r)

Erfolgt ein Zugriff auf eine Datei, wird die ACL abgearbeitet um einen Eintrag für den Nutzer zu finden. Gibt es eine Übereinstimmung so werden dem Nutzer die Rechte aus dem Eintrag in der ACL gewährt.
In einigen Implementierung werden mehr Permissions als nur read, write und excecute zur Verfügung gestellt, wie zum Beispiel delete, rename, search, etc.

4.3.3 Gerätesicherheit

Gerätetreiber (normalerweise unter /dev) werden von vielen Systemprogrammen genutzt um auf Daten zuzugreifen, die im Arbeitsspeicher, auf Festplatten, auf Diskettenlaufwerken, gespeichert sind. Die Liste von vorstellbaren Geräten ist zu lang um sie hier aufzuzählen. Unsachgemäß geschützte Geräte bieten Angreiffern eine Möglichkeit zugriff zum System zu erlangen. Deshalb gibt es ein paar Regeln die man beachten sollte:
Die Dateien /dev/kmem, dev/mem und dev/drum sollten nie von unprivilegierten Nutzern gelesen werden können. Diese Dateien sollen immer dem Root eines Systems gehören und die Gruppe kmem sollte im Modus 640 sein.
Die Disk Devices eines Systems sollten auch immer dem Root des Systems gehören und der Gruppe Operator zugeordnet sein, mit den Rechten 640. Um herauszubekommen welche Geräte im System vorhanden sind empfiehlt sich der Befehl df.
Mit wenigen Ausnahmen sollen die restlichen Devices auch dem Root des Systems gehören und im Mode 600 oder 640 sein.

4.3.4 Backups

Der Sinn und Nutzen von Backups muss wohl nicht weiter erklärt werden, sie schützen nicht nur vor komplettem Datenverlust bei einem Hardwarefehler und bei versehentlichen löschen von Dateien, sie bieten auch die Möglichkeit ein System wieder herzustellen das von einem Angreifer zerstört wurde. Ebenso hat man dank Backups eine bessere Chance die Anwesenheit eines Hackers im System festzustellen, indem man die Dateien festhält, die er später wieder löschen würde.
Die verschiedenen Unixsysteme bieten unterschiedliche Backup-Systeme. Ein gutes Backup-Programm sollte jedoch in der Lage sein, das komplette Dateisystem zu sichern und ein System so wiederherzustellen, wie es beim erstellen des letzten Backup war.
Programme wie tar und cpio sind für Backups weniger gut geeignet, stattdessen

sollte lieber dump verwendet werden. Eine gute Backupstrategie sollte min. einmal im Monat ein komplettes Backup des Systems erstellen. Partielle Backups sollten min. zweimal wöchentlich durchgeführt werden, ideal wäre natürlich täglich. Bei den Backups sollte auch für Redundanz gesorgt werden, damit im Falle eines Backupverlustes noch ein zweites vorhanden ist. Ebenso sollten alte Backups nicht gelöscht werden bevor die neuen Backups erstellt wurden.

4.3.5 Überwachen der Dateisystem Sicherheit

Auf Sicherheitsprobleme zu achten ist sehr wichtig um ein System sicher zu machen. Hauptsächlich sollte man auf Dateien achten die von unautorisierten Nutzern modifiziert werden können oder die Nutzern versehentlich zu viele Rechte gewähren oder die Angreifern unbeabsichtigt Zugang gewähren könnten. Es ist auch wichtig nicht autorisierte Änderungen am Dateisystem feststellen zu können, um diese rückgängig machen zu können. Dafür stehen in Unix-Systemen verschiedene Mechanismen zur Verfügung. Das find Kommando kann zum Beispiel genutzt werden wenn man eine bestimmte Datei sucht die man für unsicher oder gefährlich für das System hält.
Der sum Befehl kann genutzt werden um festzustellen ob eine Datei verändert wurde, dies funktioniert auch wenn die Dateigröße sich nicht geändert hat und auch das Änderungsdatum gleich geblieben ist, da sum mit Checksummen arbeitet.
Es ist zwingend notwendig regelmäßig das System auf Programme zu untersuchen die unautorisierten Nutzern zu viele Rechte einräumen (set-user-id, set-group-id) zu untersuchen, da diese ein Sicherheitsrisiko darstellen.
Ein beliebter Trick der Angreifer ist in den Superuser Account einzubrechen und ein Programm zu hinterlassen, das das set-user-id Bit für den root setzt. Um solche Dateien zu finden nutzt man etweder den find Befehl oder entsprechende Skripte.
Das Prüfen von Device Files ist ebenfalls sehr wichtig, da diese Dateien es Programmierern ermöglichen Geräte auszulesen oder auf ihnen zu schreiben. Daher sollte man regelmäßig im /dev Verzeichnis prüfen, ob ein Hacker eine eigene Devicetreiber hinterlassen hat, wenn dies der Fall ist sollte diese Datei unbedingt gelöscht werden.
Eine weitere Sicherheitslücke stellen Dateien dar die von jedem verändert werden können, besonders wenn es sich dabei um Systemdateien oder um Dateien, die bei Systemstart geladen werden, handelt. Jeder Angreifer hat dank solcher Dateien die Möglichkeit dort seine eigenen Programme mit starten zu lassen.
Eine Methode um eventuelle Trojaner, Viren, Würmer oder ähnliches aufzuspüren, ist das Suchen nach Dateien die keinen Besitzer haben. Solche Dateien sind meist ein Hinweis darauf, dass ein Hacker diese Datei im System hinterlassen hat um Schaden anzurichten. Diese Dateien spürt man mittels:

find / -nouser -print aus.

Ein sehr nützliches Mittel zur Überwachung des Dateisystems sind Checklisten. Diese können einfach über Unix-Standard-Befehle genutzt werden. Hierfür muss man nur die Befehle ls und diff nutzen. Als erstes muss man eine Master-Checkliste erstellen dies geschieht über den Aufruf:

ls -aslgR /bin /etc /usr >MasterList

Die Master-Checkliste enthält nun eine komplette Liste von allen Dateien die sich in diesen Verzeichnissen befinden. Man kann manuell Zeilen aus dieser Liste verändern oder löschen wenn man weiß, dass diese Datei demnächst sowieso geändert wird. Die Master-Checklist-Datei sollte nun irgendwo gesichert werden wo es sehr unwahrscheinlich ist, dass sie ein möglicher Angreifer findet, am besten auf einem anderen System. Um nun das Dateisystem auf Änderungen und Abweichungen zur Master-Checklist zu überprüfen muss man den oben genannten Befehl nochmals verwenden und den Output in eine andere Datei schreiben lassen.

ls -aslgR /bin /etc /usr >CurrentList
nun kann man einfach das Kommando
diff MasterList CurrentList

nutzen, Zeilen die nur in der Master-Checkliste enthalten sind werden mit einem vorangestellten kleiner-als-Zeichen (¡) ausgegeben. Zeilen die nur in der CurrentList einhalten sind werden mit einem vorangestellten gröger-als-Zeichen (¿) dargestellt. Wenn eine Datei doppelt auftaucht einmal mit einem vorangestellten kleiner-als-Zeichen und darunter gleich mit einem vorangestellten größer-als-Zeichen so bedeutet dies, dass die Datei nach der Erstellung der Master-Checkliste modifiziert wurde. Wenn man den Mechanismus der Checklisten regelmäßig einsetzt und die Master-Checkliste oft updatet, dann bietet dies eine hervorragende Möglichkeit sein Dateisystem zu überwachen. Um das Überwachen mittels Checkliste noch robuster zu machen sollte man den sum-Befehl nutzen, um für jede Datei auf der Checkliste noch zusätzlich eine Checksumme zu berechnen.

4.4 Trusted Path

4.4.1 Was ist ein Trusted Path?

Ein Trusted Path ist bekannt als ein klassischer Schutzmechanismus in einem Sicherheitsmodell. Es ist kurz gesagt eine Mechanismus zum Überprüfen, ob ein Benutzer auf die trusted Software zugreifen kann, ohne das durch andere Prozesse oder Benutzer verglichen oder überprüft werden muss ob die Software so arbeitet, wie wirklich beabsichtigt. Ein Konzept, das versucht, einen trusted Mechanismus, der den Benutzer davor bewahrt beliebigen Code auszuführen, wird normalerweise Trusted Path Execution oder TPE genannt. Es hat eine Menge Sicherheitsverbesserungen und Patches entweder für Linux oder die BSD-Kernel gegeben, die diesen Mechanismus einführen. Ausgehend von diesen Implementierungen besteht TPE aus den folgenden 3 Hauptkomponenten: Trusted Path, Trusted ACL und einer Policy.

Trusted Path

Ein Trusted Path ist ein solcher im engeren Sinne, das heißt, dass ein Path vertrauenswürdig ist, wenn der Superuser der Besitzer des Elternverzeichnisses ist und weder eine Gruppe noch alle anderen Schreibrechte für diesen Path besitzen.

Trusted ACL

Eine Trusted ACL ist eine Liste von Nutzern, welche neben dem Superuser für vertrauenswürdig erachtet werden.

Policy

Die Policy ist eine Regel, mit deren Hilfe entschieden werden kann ob ein Executable ausgeführt werden. Die Entscheidung wird vom Tusted Path und der Trusted ACL abhängig gemacht. So eine Policy könnte folgender Maßen aussehen:

- Trusted user, trusted path → Nutzer kann das Executable ausführen
- Trusted user, untrusted path → Nutzer kann das Executable ausführen
- Untrusted user, trusted path → Nutzer kann das Executable ausführen
- Untrusted user, untrusted path → Nutzer kann das Executable nicht ausführen

4.4.2 Bewertung des Unix Trusted Path Konzeptes

Hier liegt die Schwäche definitiv darin, dass ein vertrauenswürdiger Nutzer ein Executable in einem untrusted Path ausführen kann. Für einen Angreifer ist es unter Umständen nicht allzu schwierig verfälschte Programme auf einem System zu platzieren. Beispielsweise suchen Unix-Kommandointerpreter ausführbare Dateien in allen Verzeichnissen, die in der Umgebungsvariable PATH aufgelistet sind. Beinhaltet die Umgebungsvariable PATH Verzeichnisse, die nicht vertrauenswürdig sind (d.h. in denen beliebige Benutzer schreiben dürfen), dann kann ein Angreifer dort leicht Trojanische Pferde unterbringen.
Wenn die Umgebungsvariable PATH den Punkt . enthält (muss in der PATH-Variable am Anfang stehen), dann kann von einem Angreifer in einem allgemein schreibbaren Verzeichnis (z.B. /tmp) ein ausführbares ls Kommando angelegt werden. Ein Kommandointerpreter wird dann beim Wechsel in dieses Verzeichnis automatisch das neue ls Kommando ausführen.
Bedauerlicherweise verfügen die meisten Unix-Implementierungen über keinen richtigen Trusted Path der für vertrauenswürdige Executables bürgt, nicht einmal für die Login- Sequenz.
Eine Methode ist es das Drücken einer bestimmten nicht verfälschbaren Tastenkombination zu verlangen, wie z. B. bei Windows NT/2000 ßtrg-alt-entf", weil normale Programme unter Windows diesen Event nicht abfangen können, so dass dies einen Trusted Path darstellt. Es gibt ein Unix Äquivalent nämlich SSecure Attention Key"(SAK), hier wird die Tastenkombination ßtrg-alt-pauseëmpfohlen. Doch SAK ist noch sehr unreif und wir deshalb kaum oder gar nicht unterstützt. Aber SAK würde auch nicht für normale Programme funktionieren.

4.5 Netzwerk Sicherheit

In Zeiten in denen fast jeder Host mit anderen mittels Netzwerk verbunden ist, ist es essentiell dass das Unix-System Sicherheitsmechanismen für den Schutz vor Angriffen aus dem Netz bietet. Im Folgenden werden einige dieser Mechanismen vorgestellt.
Zuvor wird aber erst ein Überblick über die herkömmliche Unix Netzwerksoftware und die damit verbundenen Sicherheitsrisiken gegeben, die Software kommt nebenbei erwähnt hauptsächlich von der Berkeley Unix-Implementierung. Eines der bequemsten Features der Berkeley Netzwerk-Software ist das Trusted Hosts Konzept. Dieses erlaubt das Spezifizieren von anderen Hosts (oder auch Nutzern), die als vertrauenswürdig angesehen werden. Diese Hosts dürfen sich ohne Angabe eines Passwortes einloggen und Remote-Befehle ausführen. Natürlich birgt dieses Konzept ein Sicherheitsrisiko, so gab es Internetwürmer, die das Trusted Hosts Konzept nutzten, um sich auszubreiten. Da das Konzept aber eigentlich auch von Nutzen sein kann gibt es Mechanismen, die zur Bewahrung von soviel Sicherheit wie möglich dienen. Einen solchen Mechanismus stellt die Datei /etc/hosts.equiv dar, in ihr können Hosts, die vertrauenswürdig sind abgelegt werden. Nur Nutzer die auch einen Account auf dem System haben können sich von einem der Remote -Maschinen einloggen. Eine ähnliche Datei ist die .rhosts Datei, welche Kombinationen von Hosts und Nutzern beinhaltet, wobei jeder Nutzer für seinen Account eine .rhosts Datei anlegen kann. Dies ist ein sehr großes Sicherheitsrisiko, weil so der Superuser nicht mehr die Kontrolle hat wer auf das System zugreifen kann. Deshalb ist die hosts.equiv Datei sicherer als die .rhost Datei, denn sie kann nur vom Superuser verwaltet werden. Aber es gilt für beide, dass wenn ein Eindringling es schafft diese Dateien zu verändern, er Zugriff auf das gesamte Netzwerk (Trusted Hosts) hat.
Zwei weitere grundlegende Konzepte für Unix-Netwerke sind der File Transfer und das Senden von Emails, Beispielprogramme dafür sind ftp und sendmail. Aber auch sie sind sehr unsicher weil diese Dienste es Angreifern ermöglichen auf das System zuzugreifen, wenn sie im Besitz von Authentifizierungsdaten sind, teilweise ist ein Login auch ohne diese möglich. Aber dennoch ist der Gebrauch von den Vorgestellten Konzepten erwünscht, so dass man Mechanismen benötigt, die ein lokales Netz von dem Internet trennen und es außenstehenden Hosts gar nicht ermöglichen sich mit dem lokalen Netz zu verbinden. Ein solcher Mechanismus sind Firewalls. Eine Firewall besteht aus einer oder mehreren Hard- und Softwarekomponenten, die zwei Netzwerke koppeln und sicherstellen, dass jeglicher Verkehr zwischen den beiden Netzen durch die Firewall geleitet wird. Sie realisiert eine Sicherheitsstrategie, die Zugriffsrestriktionen und ggf. Protokollierungs- sowie Authentifizierungsanforderungen umfasst. Die Firewall leitet nur diejenigen Datenpakete weiter, die diese Strategie erfüllen,und führt Authentifizierung sowie ein Auditing gemäß den festgelegten Anforderungen durch.
Ein weiterer Schutzmechanismus sind so genannte Paketfilter, diese werden auch von Firewall -Software genutzt. Ein Paket-Filter inspiziert Datenpakete, bevor sie weitergeleitet werden, und kontrolliert so den Datenfluss zu und aus einem Netzwerk. Die Filterregeln basieren auf Paketattributen wie Quell- und Zieladresse, Transportprotokoll, Portnummern und Kontrollflags. Eine der einfachsten Me-

thoden das System vor unerwünschten Netzwerkverbindungen zu schützen sind Routing Tabellen. In diesen können Netzwerke spezifiziert werden von denen und zu denen Pakete empfangen bzw. gesendet werden. Somit kann man nicht vertrauenswürdige Netze einfach entfernen und so speziellen Hosts, Netzwerken oder Unternetzen den Zugriff auf das eigene Netz verweigern. Allerdings implementieren die meisten Firewall- Systeme diesen Mechanismus ohnehin. Abbildung 1 zeigt eine Firewall mit einem Paket-Filter.
Firewalls bergen allerdings die Gefahr in sich, dass je komplexer die Konfiguration der Firewall ist auch die Wahrscheinlichkeit für Fehler in der Konfiguration zunimmt.Darüber hinaus können in Firewall-Software fehler stecken.
Unix bietet des Weiteren auch die Möglichkeit den Netzwerkverkehr zu überwachen. Alle aktiven Verbindungen lassen sich mit dem netstat- Programm beobachten. Der FTP-Demon ftpd ist sehr nützlich um FTP-Verbindungen zu loggen, mittels Option -l veranlasst er das loggen der Verbindungen durch das syslog-Programm. Die einzelnen Unix-Implementierungen bieten auch die Möglichkeit den Netzwerkverkehr direkt zu analysieren, dazu stellen sie verschiedenste Analyse- Tools zur Verfügung, als zwei Beispiele seien an dieser Stelle etherfind und tcpdump aufgeführt.

4.6 Secure Shell

Die SSH (Secure Shell) ist ein Programm mit dessen Hilfe man sich über ein Netzwerk auf anderen Computern anmelden, auf entfernten Computern Befehle ausführen und Dateien zwischen Computern übertragen kann. Es bietet „starke“ Authentifizierung und sichere Kommunikation über unsichere Kanäle. Es ist konzipiert als Ersatz für rlogin, rsh und rcp.

4.6.1 Wozu SSH?

Der gesamte Datenverkehr im Internet basiert auf dem TCP/IP Protokoll. In der momentan weltweit eingesetzten Version von IPv4 werden jedoch alle Daten unverschlüsselt übertragen. Die auf IP aufbauenden Protokolle treffen keine weiteren Schutzmaßnahmen zur Sicherung der Kommunikation. Sämtliche Kommunikation kann also abgehört werden, da die Übertragung über mehrere Computer stattfinden kann. An jedem dieser Punkte, besonders an wichtigen Knotenpunkten, ist es möglich durch sogenannte „Sniffer“Programme den Datenverkehr nach bestimmten Kriterien automatisiert zu durchsuchen. Da ebenso Passwörter wie Email unverschlüsselt übertragen werden, ist es also leicht möglich, Zugang zu Kennungen zu erhalten, Emails mitzulesen oder Kreditkartennummern abzufangen.
Dies ist besonders bei Unternehmen kritisch, da leicht Betriebsgeheimnisse in die Hände von Konkurrenzunternehmen oder Geheimdiensten gelangen können. Ein gutes Beispiel hierfür ist das Echolon-System, welches sämtliche Telefon- und Internetkommunikation nach bestimmten Stichwörtern automatisiert abhört und Informationen sammelt. Aber auch der Staat könnte sich einen Überblick über die Gewohnheiten und Vorlieben jedes einzelnen Bürgers machen.
Eine weitere Gefahr liegt in der „Echtheit“ der Verbindung. Woher weiß man,

dass gerade mit diesem Rechner oder diesem Benutzer kommuniziert wird, oder ob die Identität nur vorgetäuscht ist. Die Eindeutigkeit eines Rechners ist durch die IP-Adresse gegeben, jedoch kann jemand eine falsche IP-Adresse vortäuschen (IP-Spoofing) und sämtliche Kommunikation dorthin abfangen. Ebenso können die Daten dann manipuliert werden und falsche Informationen weitergesendet werden (Man in the middle attack). Es ist auch möglich bestehende Verbindungen auf einen anderen Rechner zu entführen (Hijacking). Einem normalen Benutzer fallen solche Manipulationen nicht auf.

4.6.2 Leistungsmerkmale der SSH

Wie schützt man sich also vor solchen Angriffen ?
Man benötigt ein System, das einen vor zwei verschiedenen Angriffsarten schützt. Die SSH gewährleistet die Vertraulichkeit der Daten durch „starke“ Verschlüsselung. Dies bedeutet, dass es nicht ohne weiteres möglich ist, die Daten zu entschlüsseln, wie z.B. durch Analyse des Quelltextes. Eine „starke“ Verschlüsselung muss einem Angreifer einen enormen Rechenaufwand zur Entschlüsselung entgegenstellen. Nur an den beiden Endpunkten der Kommunikation können die Datenpakete ver- und entschlüsselt werden. Die Authentizität der Informationen wird gesichert, d.h. jeder Endpunkt kann sich sicher sein, dass die Identität des anderen gewährleistet ist. Eine Verschlüsselung findet schon vor der Authentifizierung statt, somit werden auch keine Passwörter unverschlüsselt übertragen.
Die SSH bietet aber noch weitere interessante Möglichkeiten. Das Display eines entfernten Computers kann auf den lokalen Rechner umgeleitet werden. So ist es auch mit einem Windows-Client möglich UNIX-Programme auf dem lokalen Bildschirm zu benutzen. Port-Anfragen können weitergeleitet werden, wodurch man z.B. Emails sicher abrufen kann. Diese Technik wird als "Tunneling"bezeichnet. Ein lokaler Proxy-Server wird gestartet, der auf Anfragen an einen lokalen Port wartet. Eine solche Anfrage wird dann an den Port der entfernten Maschine über die SSH-Verbindung verschlüsselt weitergeleitet. Dies ist bei allen TCP/IP-Diensten wie HTTP, Telnet, POP3 oder SMTP möglich.
Bei langsamen Verbindungen kann es vorteilhaft sein, den kompletten Datenstrom zu komprimieren. Dies kann ebenso transparent geschehen wie die Ersetzung der Programme rlogin, rsh und rcp.
Wichtig ist aber, dass auf beiden Endpunkten der Kommunikation SSH eingesetzt wird, ansonsten läuft auf einer Teilstrecke der Datenstrom unverschlüsselt weiter. Auf einem POP3-Server kann also nur die Email verschlüsselt abgerufen werden, wenn dort ein SSH-Daemon aktiv ist.

4.6.3 SSH Verschlüsslung

Eine Anwendung ist nur so sicher wie die darin eingesetzten Algorithmen. Da die grundlegenden Funktionen der SSH auf der Verschlüsselung und der Authentifizierung beruhen, müssen diese auf ihre Sicherheit besonders kritisch analysiert werden.
Folgende Verschlüsselungsalgorithmen werden von der SSH zur Verfügung gestellt:

IDEA, DES und 3DES, weiterhin unterstützt werden Blowfish, Twofish und Arcfour.

4.6.4 Authentifizierung

SSH unterstützt drei Arten der Authentifizierung, diese sind Passwort-Authentifizierung, Authentifizierung über den Hostnamen und Authentifizierung über Publikkey-Verfahren.

4.6.5 SSH-Implementierungen

Es gibt zwei bedeutende SSH-Implementierungen die angeboten werden:

SSH 1.x , SSH 2.x (http://www.ssh.org)

SSH 1.x ist für die meisten Betriebssysteme wie UNIX-Derivate und Windows verfügbar. Der Einsatz ist für den kommerziellen und privaten Gebrauch an keine Lizenz gebunden. SSH 2.x ist eine Weiterentwicklung von SSH und für den kommerziellen Einsatz lizenzpflichtig. Es ist nicht zur SSH 1.x kompatibel. Das kommerzielle Programm wird von Datafellows angeboten und heißt F-Secure.

OpenSSH (http://www.openssh.com)

OpenSSH wurde von den Open-BSD Entwickelern 1999 freigegeben und ist kompatibel zur SSH 1.x. Die verwendeten Algorithmen sind frei, wodurch SSH im privaten als auch kommerziellen Umfeld ohne Lizenzierung eingesetzt werden kann.

4.6.6 Installation

Die Installation wird zunächst auf dem Server durchgeführt.

- Aufruf des Programms *configure*
  Dabei werden die Umgebungsvariablen, für das spätere Kompilieren gesetzt. (Pfad des C-Kompilers usw.) Desweiteren wird festgelegt welche Verschlüsselungsalgorithmen verwendet werden sollen (RSA, IDEA usw.).

- Aufruf von *make*
  Programmdateien werden kompiliert.

- Aufruf von *make-install*.
  Das Programm wird installiert. Dabei muss beachtet werden, dass nur bei Installation durch den Superuser es allen Benutzern ermöglicht wird SSH zu nutzen. Wenn es sich um die Erstinstallation auf dem Rechner handelt, wird der sogenannte Host-Key (privater und öffentlicher) generiert. Dieser Host-Key ist bei jeder Sitzung gleich und identifiziert den Rechner eindeutig. Wenn dieser Schlüssel verloren geht oder geändert werden muss, kann dies mit Hilfe des Kommandos *ssh-keygen2* geschehen. Es ist allerdings zu beachten, dass danach alle Clients den neuen Host-Key (nur den öffentlichen) erhalten müssen, da sie sonst den Server nicht mehr erkennen (der neue öffentliche

Host-Key steht noch nicht in der Liste der bekannten Server). Für jeden SSH-Daemon wird ein anderer Host-Key angelegt.

- Der Aufruf von sshd2 startet den SSH-Daemon. Danach kann durch den Aufruf von *ssh2 localhost* festgestellt werden ob die ausführbaren Dateien richtig arbeiten. Um die Kommunikation zu testen, muss man SSH auf einem zweiten Rechner installieren.

- Generierung der Schlüssel
  Der Aufruf von *ssh-keygen2* erzeugt einen geheimen und einen öffentlichen Benutzer-Schlüssel. Durch die Option *-P* können die Schlüssel ohne Passwort erstellt werden, wie z.B. der Host-Key. Die SSH akzeptiert auch Schlüsselpaare die mit PGP erstellt wurden.

- Einbinden der Schlüssel in die SSH
  Der öffentliche Schlüssel muss auf dem entfernten Rechner in das Benutzerverzeichnis kopiert werden, üblicherweise ins Unterverzeichnis *.ssh2* . In diesem Verzeichnis muss der Name des Schlüssels in die Datei *authorization* eingetragen werden. In dieser Datei werden alle Schlüssel eingetragen, die zur Einwahl berechtigen. Der private Schlüssel verbleibt nur auf dem lokalen Rechner. Er liegt üblicherweise im Unterverzeichnis *.ssh2* des Benutzerverzeichnisses. Der Name des Schlüssels muss in die hier ebenfalls befindliche Datei *identification* eingetragen werden. Die benötigten Schlüsselwörter können aus der Online-Hilfe entnommen werden.

4.7 Network Information System(NIS)

Das NIS (Network Information System), früher als YP (SUN Yellow Pages) bekannt, erlaubt die zentrale Verwaltung eines Verteilten Systems. Hierzu gehört die zentrale Verwaltung von Passwörtern, Gruppen, Services, Rechnernamen etc. Die Verwaltung wird über einen Domainnamen geregelt. Alle Informationen sind in einfachen Tabellen (Maps) gespeichert und über eine Reihe von Rechnern verteilt. Ein ganz wesentlicher Gesichtspunkt dabei ist, dass alle Rechner dieselben Tabellen haben. Außerdem müssen Sie zueinander sicher sein und die NIS-Informationen sollten nicht an die Außenwelt dringen können.
Alle Rechner, die dieselben NIS-Tabellen (mit demselben Inhalt) verwenden, gehören zu derselben NIS-Domaine. Innerhalb einer NIS-Domäne gibt es genau einen Master: das ist der Rechner, auf dem die Tabellen gewartet werden. Von dort werden automatisch exakte Kopien der Tabellen an die Slave-Server verteilt. Die NIS-Klienten haben keine Kopien der Tabellen, sondern fragen jedesmal bei einem der Server (Master oder Slave) nach.
Jeder Rechner (auch ein Server) bindet sich an eine Domäne mit dem ypbind-Prozeß. Dieser ruft per IP-Broadcast nach einem Server und merkt sich die Adresse des ersten Servers, der antwortet. Weitere Anfragen werden dann direkt an den entsprechenden Server geschickt. Diese Zuordnung wird mit dem Kommando ypwhich angezeigt.
Auf jedem Server (Master oder Slave) läuft ein ypserv-Prozeß. Dieser verwaltet die Tabellen (Beim Master: Original, auf den Slaves: Kopien) als Dateien auf der

Platte (in der Regel unter /etc/yp oder /var/yp. Um das Auffinden eines Eintrags schnell zu machen, werden die Tabellen als .dbm-Dateien gehalten, und zwar meist in /var/yp/Domainname/Tabellenname.
Der NIS birgt ein hohes Sicherheitsrisiko, denn die zentrale Verwaltung wird über einen Domainnamen geregelt. Wenn dieser Domainname bekannt ist und eine unsichere Version des NIS-Dämons verwendet wird, kann jeder Rechner die verwalteten Informationen wie z.B. Passwörter bekommen. Durch den Einsatz von NIS können Schutzmassnahmen wie Shadow Passwortdateien umgangen werden.
Die Folge ist, dass die Passwort-Datei /etc/passwd von jedem User innerhalb der verwalteten Rechner abgefragt werden kann. Die Original-NIS-Pakete haben oft keine Möglichkeit die Herausgabe von NIS-Informationen auf bestimmte Rechner zu beschränken. Damit ist es weltweit möglich die verwalteten Informationen abzurufen. Entsprechende Patches (welche die Möglichkeit beinhalten nur definierte Subnetze auf die Informationen zugreifen zu lassen) gibt es von dem Vertreiber (z.B. SUN). Auf Grundlage von so kopierten Passwort-Dateien kann man Passwörter erraten und sich somit Zugang zu dem Rechner verschaffen. Um diesen fatalen Folgen entgegen zu wirken müssen gute Paßwörter, die Sonderzeichen enthalten, lang genug sind und nicht einfach zu erraten sind verwendet werden außerdem ist eine regelmäßige Änderung der Passwörter notwendig.
Einige Unix-Systeme können die Passworte in speziellen Dateien ablegen, so dass normale Benutzer keinen Zugriff auf die verschlüsselten Passworte haben. Möglich ist ebenfalls eine alternative Benutzer- Authentifizierung durch Smart-Cards oder unveränderliche Merkmale (z.B. Stimme, Retina).

4.8 Das Network File System (NFS)

Das Network File System (NFS) wurde entwickelt um mehreren Hosts die Möglichkeit zu geben Dateien über das Netzwerk auszutauschen. Meist wird es verwendet um Diskless Workstations in Büros zu installieren, bei denen die Datenhaltung zentral auf nur einem Rechner funktioniert.
NFS hat standardmäßig keine Sicherheitsfeatures aktiviert, das hat zur Folge, dass jeder Host im Netzwerk Zugriff auf die Dateien via NFS hat. Es gibt jedoch einige einfache Dinge die bei der Konfiguration von NFS eingehalten werden müssen, um NFS sicherer zu machen. Seit 2003 gibt das Network Filesystem Version 4. NFSv4 implementiert verschiedene Security-Methoden. Das Protokoll verhandelt die Nutzung. Exports können einer Methode/Sicherheitsstufe zugeordnet werden. Sicherstellung von Authentifizierung, Integrität und Vertraulichkeit je Request durch sichere Remote Procedure Calls (RPCSEC_GSS API) möglich. Zur Authentifizierung wird desweiteren Kerberos V5 und LIPKEY/SPKM3 (Low Infrastructure Public Key Mechanism) genutzt.
Das Filesystem ist nicht mehr so stark UNIX-zentriert: Unix Datei Attribute: Read, Write, Execute, Sticky-Bit, SUID, SGID Windows Datei Attribute: Hidden, System, Read-Only, Archive und ein paar mehr (File Locking). NFSv4 stellt erweiterbare, nummerische Attribute zur Verfügung. Darüber hinaus werden ACLs genutzt um auf das Dateisystem zuzugreifen.

4.8.1 Die exports Datei

Die Datei /etc/exports ist der wichtigste Teil der NFS Konfiguration, in ihr wird aufgelistet welche Dateisysteme oder Verzeichnisse exportiert werden (d.h. das Dateisystem oder Verzeichnis wird für Mounting via NFS freigegeben). Eine Zeile in dieser Datei enthält den Namen des Dateisystems welches exportiert werden soll und eine Liste von Hosts oder Netzgruppen, welche das Recht haben dieses Dateisystem zu mounten. Ist hinter einem Eintrag kein Hostname angegeben so darf jeder dieses Dateisystem mounten. Da nach der Installation standardmäßig nur das Dateisystem in der ersten Zeile steht und keine Hosts dahinter angegeben sind kann jeder Host das komplette Dateisystem mounten, daher sollte diese Datei unbedingt angepasst werden.
In einigen NFS Versionen kann hinter dem Dateisystem das gemountet werden soll auch noch angegeben werden welche Rechte (Lesen, Schreiben, kompletter Zugriff) den Hosts eingeräumt werden. Wenn man Änderungen in der exports-Datei vorgenommen hat muss man noch:

exportfs -a

aufrufen, erst dann haben die Änderungen Auswirkungen.

4.8.2 Einschränkung der Superuser Zugriffserlaubnis

In den meisten Versionen von NFS hat der super-User keinen unbeschränkten Zugriff auf Dateien via NFS. Dies liegt daran, dass jede NFS Anfrage mit der Superuser-Id zu einer NFS Anfrage mit der ID Nobody wird. Im Fall, dass ein Angreifer auf einer der Client-Workstations Superuser (SU) Permissions erhält, kann dieser Angreifer dann keine Daten auslesen die er nicht auch ohne SU-Status hätte auslesen können.

4.8.3 Umgang mit Set-User-Id und Set-Group-Id Programmen

Wie bereits in Kapitel 3 erläutert wurde, können Programme bei denen das Set-User-Id- bzw. Set-Group-Id-Bit gesetzt ist, ein Sicherheitsproblem darstellen, da sie bei ihrer Ausführung besondere Privilegien gewähren. Deshalb besteht die Möglichkeit das Systemverhalten für solche Programme zu modifizieren. So kann man zum Beispiel mit dem Befehl:

mount -o nosuid filesystem mountpoint

festlegen das die Set-User-Id und Set-Group-Id Bits ignoriert werden.

4.8.4 Überwachung der NFS Sicherheit

Im Allgemeinen ist NFS durch die export Dateien bestimmt, welche festlegen wer Zugriff auf bestimmte Daten hat. Der Befehl showmount -a kann genutzt um alle Hosts anzuzeigen die momentan etwas vom Server gemountet haben. Stellt man

noch die Option -d dahinter, wird zusätzlich eine Liste von Verzeichnissen angezeigt, die von irgendeinem Host gemountet sind.
Nfswatch ist ein frei verfügbares Tool, welches dazu benutzt werden kann den NFS traffic anzuzeigen. Nfswatch kann auch genutzt werden, um eine Liste der Dateien anzeigen zu lassen, auf die Hosts zugegriffen haben. Dies ist sehr nützlich wenn man feststellen will ob ein Angreifer Zugriff zu Daten erlangen will.

4.9 Der Remote File Sharing Service

Der Remote File Sharing Service ist eine Alternative zu NFS. RFS bietet eine exakte Kopie eines Unixdateisystems, damit ist RFS auf Unixsysteme beschränkt bietet aber zusätzliche Optionen die NFS nicht hat. So kann zum Beispiel auf Remote Devices zugegriffen werden. RFS teilt die Hosts in Domänen ein, diese Domänen sind ähnlich zu denen von NIS jedoch nicht identisch. Jede Domäne hat einen primären Namens-Server und beliebig sekundäre Namens-Server. Diese Server bieten eine Liste von Servern und Ressourcen. Wenn ein Client nun eine Ressource anfordert bekommt er von einem der Name-Server mitgeteilt wo er diese Ressource findet. Es gibt vier Sicherheitsstufen, die es erlauben RFS zu schützen:

- Verbindungssicherheit
- Mount-Sicherheit
- User- und Gruppenmapping
- reguläre Unix Datei Permissions

Diese werden im nun folgenden Teil näher beschrieben.

4.9.1 Verbindungssicherheit

Wenn ein Host versucht seine erste Ressource vom System via RFS anzufordern, so versucht er eine Verbindung zu ihrem Host über das Netzwerk zu erhalten. Wenn diese Verbindung aufgebaut wurde so kann jede Ressource die zum Mounten bereit steht auch von diesem Host gemountet werden. Man kann jedoch den Zugriff einschränken indem man die RFS Verifikationsprozedur verwendet. Diese Prozedur verlangt von jedem Host ein bestimmtes Passwort bevor er eine Verbindung zu ihrem Host aufbauen darf. Diese Passwörter müssen auf ihrer Maschine manuell für jeden Host erstellt werden, mit Hilfe des Aufrufes:

rfadmin -a domainname.hostname

Nun wird für jeden Host der auf diese Art erfasst wurde beim initialen Verbindungsaufbau ein Passwort verlangt. Host die nicht erfasst wurden können sich jedoch noch immer ohne Passwortabfrage zu unserem System verbinden, um dies zu ändern muss das RFS-Startup-Skript geändert werden und um den Befehl rfstart -v erweitert werden. Dadurch werden RFS Verbindungen von unbekannten Hosts ignorieren.

4.9.2 Mount Sicherheit

Sobald eine Verbindung aufgebaut wurde kann der Host alle Dateisysteme die freigegeben wurden mounten. Welche Ordner und Ressourcen freigegeben werden kann man mittels share Kommando festlegen, dabei kann man auch festlegen welche Rechte die Hosts auf den freigegebenen Ressourcen haben.

4.9.3 User und Gruppen Mapping

RFS gibt allen Nutzern eine besondere Gast-Nutzer-Id, dies sichert einen maximalen Level an Sicherheit, weil so kein Nutzer besondere Rechte erhält. Das ist meist sehr unpraktisch da Nutzer hierdurch zum Teil nicht auf ihre eigenen Dateien zugreifen können. Deshalb bietet RFS die Möglichkeiten des Mapping von Remote User und Gruppen Ids. Um diese zu konfigurieren muss man die Dateien uid.rules und gid.rules editieren.

4.9.4 Überwachen der RFS Sicherheit

Der Befehl idload kann genutzt werden um aktuelle Nutzer und Gruppenmappings anzuzeigen. Es sollte ihnen bewußt sein, dass nur Regeln für Systeme angezeigt werden, die momentan gemountet sind. Der Befehl dfshares kann genutzt werden um alle Ressourcen anzuzeigen die für das Mounting via RFS freigegeben sind. Es wird auch angezeigt welche Server diese Ressourcen anbieten und welche Rechte für die Ressourcen bestehen. Als letztes kann der Befehl dfmounts genutzt werden um festzustellen welche Remote-Hosts momentan Ressourcen gemountet haben, der Pfad wird hierbei ebenfalls angezeigt.

4.10 Schutz vor Superuser-Attacken

Die Sicherheitsmechanismen des Unix-System werden fast alle hinfällig, wenn der Superuser versucht auf Daten und Programme der Nutzer zuzugreifen. Obwohl der Superuser fast alle Rechte hat, gibt es doch ein paar Dinge, die er nicht kann. Diese Dinge sind:

- Änderungen an Dateisystemen vornehmen, die nur die Leseberechtigung haben
- Die Passworte der Datei /etc/passwd entschlüsseln
- Prozesse abbrechen, die in einer Warteschleife stecken
- Verschlüsselte Dateien ohne Passwortangabe entschlüsseln

Diese Schwächen sind der Ansatzpunkt zum Schutz vor Superuser-Attacken. Obwohl der Superuser Dateisysteme mit Leseberechtigung nicht bearbeiten kann, so kann er sich jedoch als der jeweilige User anmelden, da er kein Passwort zu Anmeldung benötigt, und die Rechte neu zu verteilen. Daher braucht er auch nicht die /etc/passwd entschlüsseln und somit ist dies auch kein Schutz vor Superusern. Außerdem kann der Superuser Prozesse, die sich in einer Warteschleife befinden, nicht

unterbrechen. Er kann jedoch das System zum Absturz bringen und somit alle Prozesse beenden. Der einzige effektive Schutz vor Superusern ist die Verschlüsselung von Daten, wobei die Kriterien für sichere Passworte eingehalten werden sollten. In Unix gibt es einen Befehl, mit dem man Dateien verschlüsseln kann. Dieses Kommando lautet crypt. Leider ist dieser Algorithmus sehr unsicher, sodass er mit einem entsprechenden Programm geknackt werden kann. Aber es gibt bereits Unix-Implementierungen wie z.B. Trusted Solaris, die die Rechte des Superusers auf verschiedene Superuser aufteilen, d. h. es werden Rechte nach Rollen verteilt. Diese Rollen sind bei Trusted Solaris der Security Administrator, der System Administrator und der Operator. Darüber hinaus werden alle Superuseraktivitäten überwacht, d. h. die Superuser kontrollieren sich gegenseitig.

4.11 Schlussfolgerung

Zur Sicherung eines Accounts sind gut gewählte Passworte das A und O. Sie stellen den wohl wichtigsten Aspekt bei der Sicherung dar und sollten nicht unterschätzt werden. Sie haben die entscheidende Rollen beim Login und bei der Verschlüsselung wichtiger Daten. Um jedoch ein höheres Maß an Sicherheit zu erhalten sollten diese in regelmäßigen Abständen geändert werden. Außerdem sollten die Zugriffsrechte der eigenen Dateien beschränkt sein und ebenfalls in regelmäßigen Abständen nach gefährlichen Konfigurationen wie SUID-Rechten suchen. Wenn möglich sollten noch ëxterneSSicherungsmöglichkeiten wie z.B. eine Firewall installiert werden. Diese Vorkehrungen sollten vor unbefugten Zugriffen schützten. Jedoch sollte man sich darüber im Klaren sein, dass es zur Zeit keine absolute Sicherheit geben kann und dass ein fähiger Angreifer früher oder später in jedes System kommt. Das Unix-Sicherheitskonzept basiert auf sehr vielen Konfigurationsdateien, so dass der Aufbau einer Systemsicherheit einen hohen administrativen Aufwand bedeutet. Mittlerweile gibt es aber auch viele Tools (z.B. Suse YaST) und Skripte, die für etwas Erleichterung sorgen. Wobei hier wieder die Frage nach der Vertraulichkeit dieser Software gestellt werden sollte. Abschließend kann man sagen, dass Unix nicht unbedingt eines der leicht-konfigurierbarsten, nutzerfreundlichsten Sicherheitskonzept bietet.

Literaturverzeichnis

[1] http://www.linuxsecurity.com.

[2] The Security Portal for Information System Security Professionals. http://www.infossec.net/infossec/unixsex1.htm.

[3] David A. Curry. *A Guide for Users and Administrators*. Addision-Wesley Professional Computing Series, 2 edition, April 1996.

[4] Kurt Seifried. LINUX Administrator's Security Guide. http://www.secinf.net/unix_security/, Oktober 2002.

[5] David A. Wheeler. Secure Programming for Linux and Unix HOWTO. http://www.bigwebmaster.com/General/Howtos/Secure-Programs-HOWTO/trusted-path.html, März 2003.

5 Betriebssysteme die erweiterte Konzepte der Verwaltung und Durchsetzung von Rechten und Politiken implementieren

René Freitag, Ronny Müller-Kuhle

5.1 Einleitung

Die UNIX-Derivate Linux und BSD gelten allgemein als sichere Betriebssysteme. Dies ist aber nur eine Halbwahrheit. Einzelplatzrechner, die immer nur von einem Benutzer bedient werden, sind mit den Standartinstallationen meist gut gesichert. Kritisch wird es aber schon, wenn viele Benutzer mit unterschiedlichen Anwendungsaufgaben an einem Rechner arbeiten. Hier reicht oft die normale Aufteilung nach Administrator, User oder auch Superuser nicht aus. In modernen Anwendungsszenarien sind feinkörnigere Einteilungen gefragt.
Ein weiteres Einsatzgebiet von Linux und BSD sind Server. Hier gelten noch weit höhere Anforderungen an die Sicherheit. In Zeiten, in denen sich jeder Laie ein Toolkit zum Hacken aus dem World Wide Web laden kann, müssen Server viel stärker gesichert werden. Die meisten Angreiffer verschaffen sich mit Hilfe von buffer overflows Zugriff auf die root-Rechte eines Servers. Sollte auf dem Server gleichzeitig ein Webserver und ein Datenbankserver liegen, hat ein Hacker bein Angriff auf den Webserver auch sofort Zugriff auf den Datenbankserver.
Zusammenfassend kann man also folgende drei Aspekte als Hauptschwachpunkte von Linux und BSD ausmachen:

- Geringe Granularität
- Discretionary Access Control
- Allmächtiger Root

Die folgende Arbeit soll einen groben Überblick über die aktuellen Arbeiten zum Thema Sicherheit bei Linux und BSD geben. Die zentralen Themen sollen SELinux und TrustedBSD sein. TrustedBSD ist aber nicht als einziges sicheres BSD zu sehen. Die meisten Elemente von TrustedBSD tauchen meist nach kurzer Zeit auch in den anderen BSD Varianten wie FreeBSD oder OpenBSD auf.
Danaben soll auch RSBAC betrachtet werden. RSBAC ist eine sehr umfangreiche Kernelerweiterung für Linux, die eine Vielzahl von Sicherheitsmerkmalen implementiert.

5.2 Security Enhanced Linux

5.2.1 Entwicklung von SELinux

Security Enhanced Linux ist eine gemeinsame Entwicklung der National Security Agency (NSA) und der Secure Computing Corporation (SCC). Ziel der Entwicklung war es, zu beweisen, dass es möglich sei, starke systembedingte Zugriffskontrolle in ein weitverbreitetes Betriebssystem einzubauen. Bemerkenswert ist hierbei, dass die NSA dazu das freie Betriebsystem Linux verwendet hat. Ausserdem stehen die Quelltexte jedem offen. Daher ist Misstrauen gegenüber dieser Organisation wohl kaum zu begründen.
Bereits im Jahr 2000 war eine erste Version von SELinux verfügbar, damals allerding noch als Kernelpatch. Mit der Vorstellung von SELinux auf der Linux Kernel Summit 2001 wurde Linus Torvald zur Entwicklung eines Security-Frameworks für Linux inspiriert. Die Linux Security Modules waren geboren. Fortan wird SELinux als Modul für Linux Security Modules (LSM) weiter entwickelt.

5.2.2 Die Flask Architektur

Als Ergebniss der Zusammenarbeit zwischen NSA und SCC stand die Flask Architektur. Ziel war es, Sicherheitsrichtlinien zu entwickeln, die möglichst flexibel sind. Der Grund dafür war, dass verschiedene Nutzer immer auch verschiedene Sicherheitsbedürfnisse haben.

Die Flask Architektur besteht grob aus drei Komponenten: Der Security Server, der Objectmanager und der Access Vector Cache (AVC). Abbildung 5.1 zeigt schematisch den Aufbau der Architektur.

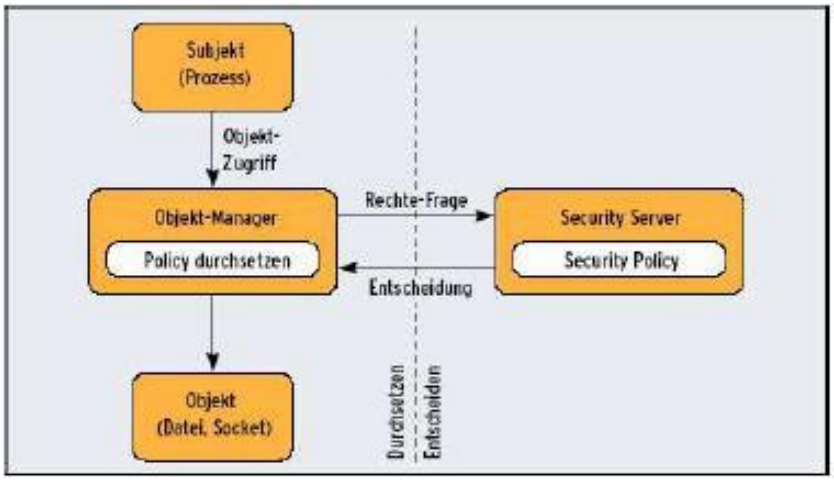


Abbildung 5.1: Schematischer Aufbau der Flask Architektur (Quelle: www-spies.in.tum.de)

Der Security Server ist dafür zuständig, zu entscheiden, ob ein Subjekt die nötigen Rechte besitzt, um auf ein Objekt zuzugreifen. Er fragt anhand der entsprechenden Security Identifier (SID) des Subjekts und des Objekts den in den Sicherheitsrichtlinien festgelegten Security Context ab. Dieser bildet die Entscheidungsgrundlage. Ein SID ist eine für jedes Subjekt und jedes Objekt eindeutige Nummer, die nur für diesen einen Rechner gültig ist. Ausserdem gelten sie nur zur Laufzeit. Mit einem Systemneustart werden die SIDs neu vergeben. Nur die Persistent SIDs (PSID) werden gespeichert und gelten auch nach einem Neustart. PSIDs sind zum Beispiel für Dateien notwendig. Nur der Security Server kann SIDs oder PSIDs einem Security Context zuordnen. Dieser Context besteht aus drei Attributen: Der Benutzeridentität, der Rolle des Benutzers und einem Typ. Hierzu mehr in Zusammenhang mit der Policy Sprache von Linux.
Die Object Manager dienen der Umsetzung der Entscheidungen des Security Servers. Die Manager übernehmen ausserdem die Aufgabe des Markierens der Subjekte und Objekte mit den SIDs.
Die dritte Komponente der Flask Architektur, der Access Vector Cache dient der Verbes- serung der Systemleistung. Die ständigen Anfragen der Object Manager an den Security Server sollten optimiert werden. Dazu wurde der AVC zwischen Manager und Server geschoben. Anfragen der Manager gehen an den Cache. Dieser prüft, ob die Anfrage schon einmal gestellt wurde und gibt dann entweder die gespeicherte Antwort zurück oder holt sich die Antwort vom Server und speichert und gibt sie weiter. (Abbildung 5.2)

Abbildung 5.2: Funktionsweise des Access Vector Cache (Quelle: wwwspies.in.tum.de)

5.2.3 Die Architektur von Linux Security Modules (LSM)

Mit der Vorstellung von SELinux war Linus Torvalds gezwungen, über eine Schnittstelle zur Sicherheitserweiterung von Linux nachzudenken. Für ihn war aber von vorherein klar, dass diese Schnittstelle universell sein müsse. Das heisst, dass Module beliebig ausgetauscht werden können. Das System sollte einfach und effizient sein und nur wenig in das System eingreifen.

Implementation von LSM

Ein Zugriff eines Subjektes auf ein Objekt läuft bei LSM wie folgt ab: Zuerst wird wie bisher die benutzerbestimmte Zugriffskontrolle durchgeführt. Erst wenn diese einen Zugriff gestattet, findet eine zweite Überprüfung statt. Dazu wird die so genannte Hook-Funktion aufgerufen, die die eigentliche Anfrage an das jeweilig geladene Sicherheitsmodul startet. Hook-Funktionen werden hierbei in zwei Kategorien aufgeteilt. Es gibt Hooks zur Verwaltung der neuen Sicherheitsfelder und Hooks zur Zugriffkontrolle. Die Hooks stellen somit ein Equivalent zu einem Treiber für Hardware dar. Der Entwickler muss nur die Hook-Funktionen implementieren, um unterschiedliche Module zu verwenden. Da vor den Hooks immer erst die Standartkontrolle durchgeführt wird, werden teilweise gar keine Hooks aufgerufen, was Performancevorteile gegenüber anderen Ansätzen bringt.
Einige Kernelobjekte, wie zum Beispiel Inodes oder Netzwerksockets wurden im Zuge von LSM um Sicherheitsfelder erweitert. Diese erlangen aber nur beim Laden eines Sicherheitsmoduls an Bedeutung.

Laden eines Moduls

Sicherheitsmodule können sowohl statisch als auch dynamisch zur Laufzeit geladen werden. Dies geschieht mit den Befehlen register_security() und unregister_security().

5.2.4 Interne Architektur von SELinux

SELinux besteht aus fünf Elementen: Der Security Server, der Access Vector Cache, das Persistent Label Mapping, neue Systemaufrufe und die implementierten Hook-Funktionen.
Wie bereits erwähnt, setzt SELinux die Flask Architektur um. Der in SELinux umgesetzte Security Server ist eine Mischung aus Role Based Access Control (RBAC) und Type Enforcement (TE). Der AVC ist wie unter 3.2 beschrieben für die Performance eingeführt worden.
Das Persistent Label Mapping übernimmt die Etikettierung eines beständigen Objektes. Einer Datei (oder einem anderen persistenten Objekt) wird somit eine PSID zugeordnet. Dazu werden in einer Datei die Abbildung von Inodes auf PSIDs gespeichert. Dadurch können neuere Dateisysteme leichter unterstützt werden. Allerdings leidet die Performance und die Konsistenz darunter.
Zwei verschieden Arten von Systemaufrufen wurden hinzugefügt. Erstens gibt es neue Versionen bestehender Aufrufe, damit Programme auf den Security Context von Kernelobjekten und Operationen zugreifen können. Daneben existieren völlig neue Aufrufe, die Programmen die Nutzung der Schnittstelle des Security Servers ermöglichen.
Die Hook-Funktionen erfüllen die von der LSM geforderten Zwecke der Zugriffkontrolle und der Vergabe der SIDs.

5.2.5 Laden des SELinux Moduls

Beim Laden von SELinux ersetzt dieses nicht das Dummy Modul, sondern hängt dieses per Stacking hinter das SELinux Modul. Dann wird der AVC initialisiert. Dies ist sinnvoll, da ja alle Anfragen an ihn gestellt werden. Nun werden nur noch die neuen Sicherheitsaufraufe in die Systemaufruftabelle eingetragen.

5.2.6 Hook Funktionen

Die Hook Funktionen sind, wie bereits erwähnt, grundlegend für die LSM. Sie müssen immer vom aktuell geladenen Modul umgesetzt werden. Diese Umsetzung ist in jedem Modul anders, da sie verschiedenste Sicherheitskonzepte an die LSM anpassen können.
Zusätzlich zu den Hook Funktionen benötigt SELinux noch weitere Hilfsfunktionen. Die Funtionen alloc_security() und free_security() dienen der Speicherplatzreservierung und -freigabe für die Sicherheitsstrukturen.
Da SELinux zur Laufzeit gestartet werden kann, ist eine Zuordnung von Subjekten und Objekten zu einem Security Context nötig, auch wenn diese schon vor dem Start des Moduls bestanden. Dazu dienen die Vorbedingungs-Hilfsfunktionen. Diese Funktionen dienen auch der Behandlung zyklischer Abhängigkeiten. Diese treten zum Beispiel auf, wenn die Sicherheitsrichtlinien geladen werden.
Die Permission Checking Helper Functions dienen der Anfrage an den AVC. Sie sind aber nicht zwingend, da Anfragen an den AVC auch direkt gestellt werden können.
Ohne zu sehr auf einzelne Details einzugehen, soll im Folgenden der Aufbau der Hook Funktionen eingegangen werden. Die Funtionen dienen im Allgemeinen der

Verwaltung der Sicherheitsstrukturen. Solch eine Struktur besteht meist aus der ID des SELinux Moduls, einem Zeiger auf das aktuelle Objekt, einem Zeiger auf eine Liste gleicher Strukturen und dem SID des Objektes. Desweiteren ist eine Referenz auf den AVC vorhanden.
Weitere Funktionen dienen der Zuweisung und Freigabe und der Behandlung von Vorbedingungen. Auch sind Funktionen implementiert, die die Operation überwachen, welche die Hook Funktion ausgelöst hat.
Der eben erläuterte Aufbau gilt für Hook Funktionen für das Laden von Programmen, Taskoperationen und Dateisystemzugriffen. Für andere Zugriffe, wie zum Beispiel IPv4, sind separate Hook Funktionen implementiert. Eine Auflistung und Beschreibung sämtlicher Hook Funktionen findet man in [8].

5.2.7 Geänderte Programme

Die Struktur von SELinux macht einige Änderungen von Programmen und Dämonen nötig. Die modifizierten Dämonen sind login, sshd und crond. Die Modifikationen sorgen dafür, dass sie für Benutzerprozesse in den entsprchenden Security Context wechseln.
Die neuen Programme haben verschiedene Aufgaben. Manche dienen der konsistenten Umsetzung der Sicherheitsrichtlinien. Andere Programme nehmen Änderungen am Security Context einer Datei vor. Einige Programme dienen dem Wechsel zwischen Rollen und Typen (siehe 2.8).
Geänderte Benutzerprogramme wie ps und ls sind so modifiziert, dass weder das Anzeigen noch das Ändern des Security Context erlauben.

5.2.8 Aufbau des Regelwerks

Wie erwähnt setzt SELinux auf eine Mischung aus RBAC und TE. Der Flask Architektur folgend, fragt der Security Manager anhander der SIDs von Objekt und Subjekt den jeweiligen Security Context ab. Davon leitet er die Zugriffsentscheidung ab. Der Context besteht aus drei Sicherheitsattributen. Diese sind die Benutzeridentität des SELinux Users, seine Rolle und einem Typ. Der Wechsel des SELinux Users ist nach dem Login nicht mehr möglich. Es können aber mehrere normale Benutzer unter einem SELinux Benutzers zusammengefasst werden.
Rollen können von einem Benutzer gewechselt werden und zwar explizit mit newrole. Dies geschieht natürlich nur, wenn der Benutzer die Berechtigung für die neue Rolle hat.
Typ Enforcement wird abgewandelt umgesetzt. Man unterscheidet nicht zwischen Typen und Domänen. Ein Typwechsel findet automatisch statt und geschieht nach den Sicherheitsregeln.
Der Security Context wird durch den Security Server zugeordnet. Dazu wird jedem Prozess ein eigener Security Context zugeordnet. Dies geschieht aufgrund des Regelwerkes, des Vaterprozesses und der Benutzer-ID des Prozesses.
Alle Sicherheitsrichtlinien werden in Dateiengespeichert. So werden die File Contexte in .fc Dateien abgelegt. (siehe [6]).

5.2.9 Erstellen einer Sicherheitsrichtlinie

Die Erstellung eines vollständigen Regelwerkes ist sehr aufwändig, daher gibt es bereits ein vorgefertigtes Regelwerk (siehe [6]). Allerdings kann ein solches vorgefertigtes Regelwerk in keinem Fall ein eigenes vollständiges Regelwerk ersetzen, da auf jedem System andere Sicherheitsbedürfnisse bestehen.
Die Erstellung eines solche Regelwerkes ist aber nicht ohne Tücke. Für ein neues Programm muss auch immer eine Regel erstellt werden. Dazu muss man aber wissen, welche Dateien gehören zum Programm und auf welche Objekte braucht das Programm Zugriff. Nun muss man sich überlegen, wie man Dateien geeignet zu Typen zusammenfasst. Hierbei ist zu beachten, dass man die Typen sinnvoll wählt. Dateien, die nur gelesen werden, sollten nicht zum selben Typ wie Dateien zählen, auf die auch schreibend zugegriffen wird. An dieser Stelle sieht man schnell ein, dass die Sicherheit des Systems nicht nur durch dass System selbst gegeben ist, sondern auch der Benutzer dazu beitragen muss.
Für einen nötigen Typwechsel müssen ebenfalls Regeln erfstellt werden. Die geschieht mit type-transition.
Detailiert wird darauf in [9] eingegangen.

5.2.10 Mögliche Einsatzgebiete

SELinux ist aufgrund des hohen Aufwandes für Erstellung des Regelwerkes nur eingeschränkt sinnvoll für Heimrechner. Multiuser Systeme können dagegen stark vom Einsatz des SELinux Moduls profitieren, sofern das Regelwerk entsprechend aufgebaut wird. In Unternehmen kann dann garantiert werden, dass verschieden Benutzer immer nur Zugriff auf die von ihnen benötigten Daten und Programme erhalten. Ein witeres Einsatzgebiet sind Rechner, die gleichzeitig Web- und Datenbankserver sind. Verschafft sich ein Unbefugter Zugriff auf den Webserver, hat er keinen Zugriff auf den Datenbankserver.

5.3 Rule Set Based Access Control

5.3.1 Entwicklung von RSBAC

Seit dem Jahr 2000 existiert dieses Projekt. Ziel der Entwicklung war ein freies und mächtiges Access Control Framework. Es basiert auf Generalized Framework for Access Control. Die Entwicklung erfolgte unabhängig von Regierungen und Firmen und stellt eine komplette Eigenentwicklung dar. Mehr Details zu RSBAC findet man auf [5].

5.3.2 Aufbau des Rahmenwerks

Ähnlich dem Ansatz von SELinux unterscheidet man bei RSBAC nach Subjekten, Objekten und Entscheidungsanfragen. Subjekte sind als Prozesse, die im Namen von Benutzern agieren, zu verstehen. Sie führen Programme aus und binden die dazu nötigen Bibliotheken ein.
Subjekte versuchen stets auf Objekte zuzugreifen. Dies können Dateien (FILE), Verzeichnisse (DIR), oder zum Beispiel Sockets (NETOBJ) sein.
Die Art und Weise, wie ein Subjekt auf ein Objekt zugreifen will, wird als Anfragetyp abstrahiert. Danach kann dann die konkrete Entscheidungsanfrage an die Entscheidungskomponente gestellt werden.

5.3.3 Entscheidungsmodule

RSBAC implementiert ein Vielzahl von Entscheidungsmodulen und Regelsätzen. Neben Mandatory Access Control (MAC) und Access Control Lists (ACL) sind noch weitere Modelle enthalten.
Functional Control (FC) erlaubt es nur Sicherheitsbeauftragten Zugriffe auf Sicherheitsinformationen und nur Administratoren Zugriffe auf Systeminformationen.
Mit Security Information Modification (SIM) kann sichergestellt werden, dass als Sicherheitsinformation gekennzeichnete Daten nur von Sicherheitsbeauftragten geändert werden können.
Auch den deutschen und europäischen Datenschutzrichtlinien kann RSBAC mit Hilfe des Privacy Models (PM) gerecht werden.
Malware Scan (MS) prüft Dateien beim Lesen und Ausführen auf bösartige Software. Dazu kann aktuell die AntiVirus Software F-Prot eingesetzt werden. Alerdings ist die Unterstützung weiterer Scanner geplant.
Mit File Flags (FF) ist es möglich Dateien und Verzeichnissen vererbbare Attribute zu vergeben. Dies können zum Beispiel read-only oder no-execute sein.
Eine der Hauptschwachpunkte bei Linux ist root. Der allmächtige root wird für viele Administrationsaufgaben benötigt. Allerdings bleibt die Frage, warum ein Datenbank-Administrator unbedingt Zugriff auf die Linux Systemdateien benötigt. Durch Role Compatibility (RC) können besonders auf Serversystemen die Rechte von Nutzern feiner verteilt werden. Alle Benutzer werden in Rollen eingeteilt. Ausserdem wird für jede Rolle und für jedes Objekt der Zugriff separat festgelegt. So kann man zum Beispiel die Administratoraufgaben verteilen.
Ein weiteres Modul ist Authentication Enforcement (AUTH). AUTH kontrolliert change owner Anfragen. Durch wird sichergestellt, dass sich Prozesse nur die Rech-

te verschaffen, die ihnen erlaubt wurden.
Linux Capabilities (CAP) sind eine Möglichkeit, Serverprogramme ohne root Rechte laufen zu lassen. Dazu werden die aufgeteilten root Rechte für jeden Benutzer und jedes Programm festgelegt.
Weitere Module sind JAIL zum Einsperren von Prozessen in chroot-Käfigen, Linux Resources (RES), zur Festlegung von Resourcenschranken und PageExec (PAX) zum Schutz vor eingeschleustem Programm-Code (siehe [1]).

5.3.4 Architektur und Ablauf der Zugriffskontrolle

Die Architektur von RSBAC (Abbildung 5.3) kann in mehrere Teile zerlegt werden. Ein Subjekt stellt dabei stets Anfragen an die Access Control Enforcement Facility (AEF) bezüglich eines Objekts. Diese fragt bei der Access Control Decision Facility (ADF) nach, ob der Zugriff gestattet wird. ADF entscheidet anhand der Entscheidungsmodule und liefert dann ein granted oder ein not granted an die AEF zurück. Diese wiederum erlaubt dann dem Subjekt den Zugriff auf das Objekt oder verweigert diesen Zugriff. Da ADF auf verschiedene Entscheidungsmodule zugreift, wird bereits bei einer negativen Entscheidung auf not granted entschieden.

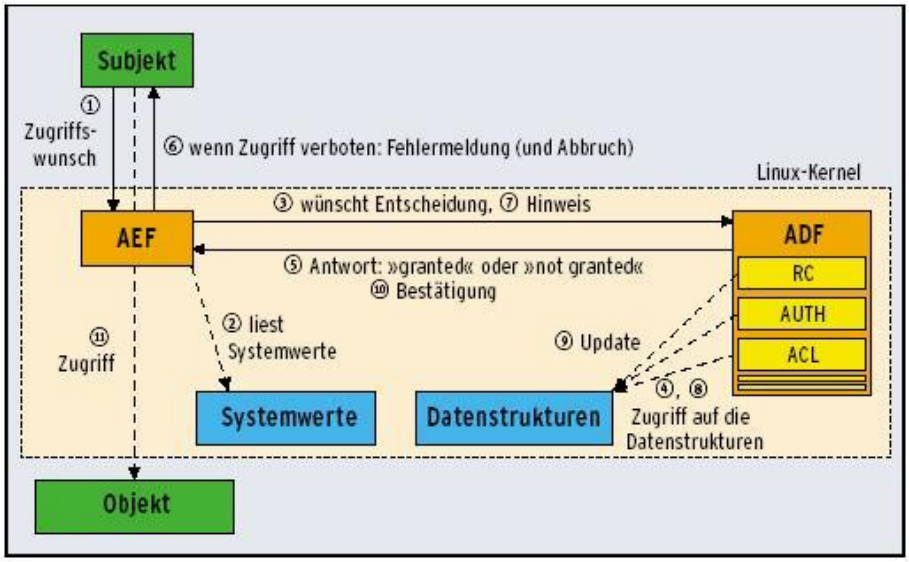


Abbildung 5.3: Aufbau der Architektur und Zugriff auf ein Objekt (Quelle: rsbac.org)

5.3.5 Installation

RSBAC ist ein Kernelpatch. Entweder man nutzt seinen vorhanden Kernel und patcht diesen oder aber man benutzt vorgepatchte Kernel. Zu Testzwecken steht auch der sogenannte SoftMode zur Verfügung.

5.3.6 Aktuelle Implementationen

Adamantix

Adamantix ist eine RSBAC Implementation auf Debian Basis. Adamantix ziel besonders auf den Servermarkt ab, da zusätzlich zu RSBAC einige besondere Netzwerkfähigkeiten implementiert wurden. Adamantix unterstützt VPN und kann als Wireless Lan Basis Station eingesetzt werden. Es bringt eine Microsoft kompatible PPP Verschlüsselung mit und kann besonders gut von Internet Service Providern eingesetzt werden. (siehe [2]).

Sniffix

Sniffix wurde an der Budapester Universität erstellt. Dazu wurde eine vorhandene Knoppix CD verändert und mit RSBAC versehen. Mit der bootfähigen CD lassen sich auf verschiedenen Rechner verschiedene RSBAC-Rollen simulieren. Es gibt Installationen für Server und Clients. Ziel war es, RSBAC näher zu testen und Studenten die Möglichkeit zu geben, in einem kontrollierten Umfeld sniffing durchzuführen.

5.4 TrustedBSD

5.4.1 Einleitung

Man darf im Bereich der sicheren BSDs nicht nur von TrustedBSD sprechen. TrustedBSD ist vielmehr als Testwiese für neue Sicherheitsmodule zu sehen. Die meisten Sicherheirsmerkmale findet man nach relativ kurzer Zeit auch in FreeBSD oder OpenBSD. Somit kann es vorkommen, dass im folgenden die Bezeichnungen der einzelnen BSD ein wenig durcheinander geraten.

5.4.2 Geschichte

BSD

Die ursprüngliche Berkeley Software Distribution (BSD) startete die Entwicklung eines Unix-Systems für die VAX und die PDP-11 bereits im Jahr 1977. Im Laufe der Zeit kamen mehrere verschiedene Distributionen hinzu. Als Ur-BSD bezeichnet sich NetBSD, die erste BSD-Ableitung. Aus NetBSD zweigt OpenBSD ab. Zu OpenBSD gehört auch die Entwicklung von OpenSSH. Das zweite Projekt, das direkt vom Ur-BSD abzweigt, ist FreeBSD. Aus FreeBSD leiten sich einige kleine, aber sehr spezialisierte Distributionen ab: z.B. TrustedBSD, welches erweiterte Sicherheitstechnologien bietet.

TrustedBSD

Am 9. April 2000 wurde das Projekt von TrustedBSD gestartet. Es wurde initialisiert, um neue vertrauenswürdige Sicherheitsfunktionen für FreeBSD zu schaffen.

FreeBSD

FreeBSD ist ein Unix-ähnliches Bestriebssystem, welches von 386BSD 0.1 abstammt und auf folgenden Systemem verbreitet ist: Intel x86 Familie, DEC Alpha, UltraSPARC, Itanium (IA-64) and AMD64. Die Arbeiten an FreeBSD wurden November 1993 von Jordan K. Hubbard, Nate Williams, Rod Grimes und David Greenman gestartet. FreeBSD 2.0 wurde im Januar 1995 veröffentlich, die aktuelle Version 5.2.1 erschien im Februar 2004. Bei FreeBSD gibt eine stabile (STABLE) und eine aktuelle (CURRENT) Version. In der CURRENT-Version werden neue Funktionen getestet, die dann in dei STABLE-Version eingebettet werden. Da FreeBSD eine komplettes Bestriebssystem ist, wird es über CVS verwaltet. Mit Pachtsets von ScureBSD und TrustedBSD kann die Sicherheit von FreeBSD zusätzlich verbessert werden.

OpenBSD und NetBSD

OpenBSD ist ein Unix-ähnliches Betriebssytem von Theo de Raadt.
NetBSD ist das erste Open-Source BSD. Im Mai 1993 erschien die erste Version 0.8, die letzte Version wurde ist die 1.6.1 vom August 2003.

5.4.3 Das Dateisystem von TrustedBSD(FreeBSD, NetBSD...)

Allen BSD-Versionen gemeinsam ist das Dateisystem. BSD verwendet das Berkley Fast Filesystem (UFS/FFS - Fast File System), wie es auch in vielen kommerzielen Unix-Systemen zum Einsatz kommt. Ein besondere Eigenschaft diese Systems ist das Soft-updates-Verfahren. Da Schreibzugriffe auf einem UFS gepuffert und optimiert werden, ist sichergestellt, daß das Dateisystem jederzeit in einen konstanten Zustand ist. Dadurch werden Zugriffe erheblich beschleunigt und Stabilität des Systems erhöht. Auch der Umgang mit fremden Dateisysteme ist für BSD kein Problem: es kann mit Partionen von DOS (FAT und FAT32), Linux (ext2fs), Windows (NTSF) und MacOS (HFS) arbeiten.

5.4.4 Allgemeine Sicherheitselemente von BSD

FreeBSD besitzt eine Reihe von Werkzeugen und Mechnismen, die Integrität und die Sicherheit des Systems und des Netzwerkes zu gewährleisten:

- verschiedene Verschlüsselungsmechanismen, wie DES oder MD5
- KeberosIV vor Version 5.0 und Keberos5 nach Version 5.0
- Firewahlfunktion mit Paketfilter IPFW
- VPN (Virtual Private Network) mit IPsec
- OpenSSH
- TrustedBSD-MAC-Framework
- Access Control Lists ACL

5.4.5 TrustedBSD-MAC-Framework

Zu FreeBSD (ab Version 5.0) gehört ein neues kernelbasiertes Sicherheitssystem, das TrustedBSD-MAC-Framework. Unter MAC (Mandatory Access Control) versteht man die vorgeschriebene Zugriffskontrolle eines Betriebssystems. Damit lassen sich Zugriffe eines Systems beim Übersetzen des Kernels, beim Systemstart und zur Laufzeit kontrollieren. FreeBSD erlaubt es, vorgefertigte Module zu laden, die vorgeschriebene Zugriffskontrollen bereitstellen oder die Systemshärtung erhöhen. Da sich das Framework noch in der Testphase befindet, ist eine stabile Version erst in FreeBSD 5.2 zu erwarten. Das MAC-Framework muss im Kernel erst aktiviert werden, um die zusätzlichen Sicherheitfunktionen nutzen zu können.Die einzelnen Module müssen dann mehr oder weniger konfiguriert werden.

5.4.6 Die Sicherheitsmodule in FreeBSD

Biba-Richtlinie zur Sicherung der Integrität (mac_biba)

Durch die Biba-Richtlinie wird die Integrität aller Systemobjekte gekennzeichnet. Dadurch wird gewährleistet, das Objekte mit hoher Integrität von Subjekten mit niedrieger Integrität nicht verändert werden. So können Subjekte mit hoher Integrität (üblicherweise Prozesse) nicht lesend auf Objekte niedrigerer Integrität (häufig Dateien) zugreifen und Subjekte niedrigerer Integrität nicht schreibend auf Objekte höherer Integrität. Die Biba-Richtlinie muss systemweit zur Verfügung stehen und ist daher fest in den Kernel integriert.

Dateisystem-Richtlinie (mac_bsdextended)

Mit dieser Richtlinie werden die Zugriffsrechte des Dateisystems erweitert. Es könne nun Regelsätze für Zugriffe von Benutzer und Gruppen auf Systemobjekte definiert werden. In den Regelsätzen wird festgelegt, auf welche Dateien und Verzeichnisse ein Benutzer oder ein Prozess zugreiffen darf. Die Regeln speichern die Benutzer und die UID (User IDentification) und GID (Goup IDentification) der

Prozesse und beschränken dadurch den Zugriff auf Objekte von anderen Besitzern oder Gruppen. Das Modul mach Sinn für Systeme, wo der Datenaustausch von Benutzer gesteuert werden soll.

Interface-silencing-Richtlinie (mac_ifoff)

Diese Richtline verhindert die Nutzung der Netzwerkkarte vom Systemstart bis zum Zeitpunkt ihrer Aktivierung. Damit wird verhindert, daß die Netzwerkkarte in dieser Zeit auf eingehende Pakette ungewollt antwortet.

Low-Watermark Mandatory Access Control (mac_lomac)

Auch diese Richtlinie befasst sich mit der Integrität der Objekte, ähnlich der Biba-Richlinie. Es können hier aber Sybjekte mit hoher Integrität leseend auf Objekte mit niedriger Integrität zugreifen. Die Integrität des lesenden Subjekt wird heruntergesetzt, damit es nicht mehr schreibend auf Objekte mit hoher Integrität zugreifen kann.

Multi-Level-Security Richtlinie (MLS, mac_mls)

Die Richtlinie bietet systemweit hierarchische und nicht-hierarchische Kennzeichen zur Markierung der Vertraulichkeit von Objekten. Es wird so garantiert, daß vertraulische Daten nicht unberechtigt weitergeleitet werden. Für die Zugangsberechtigung unterscheidet man hierarchische und nicht-hierarchische Kennzeichen, also für Verschlußsachen (Einteilung in ßtreng geheim", "geheim", usw.) und Verwirklichung des Prinzips "Kenntnis nur, wenn nötig"(need to know). Da die Richtlinie fest im Kernel eingebunden wird, müssen vorher alle Systemobjekte gekennzeichnet werden.

Rumpf-Richtlinie (mac_none)

Diese Rumpf-Richtlinie stellt nur ein Beispiel für Entwickler dar, die alle benötigten Funktionen, ohne die Zugriffsrechte im System zu verändern.

Partitions-Richtlinie (mac_partition)

In der dieser Richtline wird die Sichtbarkeit von Prozessen eingeschränkt. Den Prozessen wird eine Partitionsnummer zugewiesen und somit können diese Prozesse nur alle anderen Prozesse in derselben Partition sehen. Ohne Partitionsnummer sind alle Prozesse im System sichtbar.

See Other Uids (mac_seeotheruids)

Die See Other Uids schränkt ebenfalls die Sichtbarkeit von Prozessen ein. Hier ist aber die Sichtbarkeit anderer Prozessen von den Berechtigungen eines Prozessen abhänging. Es kann konfiguriert werden, daß diese Richtlinie für bestimmte Benutzer und Gruppen nicht gilt.

Test-Richtlinie (mac_test)

Die Test-Richtlinie stellt einen Regressions-Test für das MAC-Framework bereit. Sollten die Prüfung auf korrekte Kennzeichen fehlschlagen, so wird ein Systemstopp durchgeführt.

Access Control List (ACL)

FreeBSD bietet ab der Version 5.0 Zugriffskontrolllisten (Access Control List). Mit Zugriffskontrolllisten werden die normalen Zugriffsrechte eine UNIX-Systems erweitert. Sie ermöglichen eine feiner granuliert Sicherheitsmechanismen. Die Zugriffskontolllisten für das Dateisystem werden im Kernel aktiviert. ACL steht bei UFS1 als Erweiterung bereit und ist integriert im UFS2.

5.5 Mögliche Einsatzgebiete

Für die Frage, welches Betriebssystem man einsetzen möchte, sollte man sich vorher genau über den Einsatzzweck im Klaren sein. Möchte man ohne große Administrationsarbeit sein System sicherer machen, so ist es jedenfalls sinnvoll, eine fertige Distribution wie Adamantix mit vorgefertigten Politiken zu installieren. Dies erleichtert den Einstieg in das Feld der gehärteten Systeme.
Will man nur seine Dienste sicher laufen lassen, so sollte man eher auf ein System mit installiertem systrace zurückgreifen, da die Policy-Sprache von systrace hierzu besonders gut geeignet ist.
Will man allerdings Benutzer in ihren Zugriffen untereiander und auf das System einschränken, kann man zu allererst zu virtuellen Betriebssystemen zurückgreifen. Jedem Benutzer wird sein eigenes virtuelles System zur Verfügung gestellt. Dieses System läuft dann in einem sicheren Bereich innerhalb des Wirtsbetriebssystems. Das kostet allerding erheblich Ressourcen, so dass man eher auf die vorgestellten Systeme SELinux, RSBAC oder TrustedBSD zurück greifen sollte.
Für den Schutz streng geheimer Daten bietet sich zur Zeit ganz besonders SELinux an, da es von der NSA gerade zu diesem Zwecke entwickelt wurde.

5.6 Zusammenfassung

Einen Vergleich der einzelnen Systeme zu führen gestaltet sich ein wenig schwierig. Alle Systeme sind bemüht, die Schwächen der Linux- und BSD-Systeme auszumerzen. Sie setzen alle auf ein Mischung verschiedener Entscheidungsmodule wie RBAC und TE. Von der technischen Seite aus sind also alle Systeme geeignet, einen Server abzusichern. Bleibt allerdings die Frage offen, inwieweit die Systeme administrierbar sind. Bereits bei SELinux zeigt sich der erhebliche Aufwand für die Erstellung der Regelsätze. Somit ist anzunehmen, dass bei der Fülle der Module in RSBAC und BSD der Aufwand für die Administration erheblich steigt. Bei einer Entscheidung, ob man auf ein gehärtetes System umsteigen will, sollte man sich also der bevorstehenden Arbeit bewusst sein. Wenn man nur Standartregelsätze verwendet, verfehlt man den Gedanken der gehärteten Systeme. Denn nur die höchstindividuellen Einstellungsmöglichkeiten unterscheiden diese Systeme vom Standart. Man sollte sich also im Vorraus über die Tools zur Administration informieren und daran auch eine Entscheidung treffen.
Linux und BSD lassen sich umfassend absichern. Leider geht erhöhte Sicherheit mit einem drastisch erhötem Maße an Arbeitsaufwand einher. So hat man nicht nur mit der Einstellung des Systems einiges zu tun. Auch die Planung davor nimmt Zeit in Anspruch. Man muss sich schon bewusst sein, wie man Benutzer einteilt und welche Programme welche Ressourcen benötigen.
Abschliessend kann man sagen, dass es durchaus Möglichkeiten gibt sein System gegen Angriffe zu schützen. Auch muss man für die Software ansich kein Geld ausgeben. Allerdings kann der erhöhte Arbeitsaufwand schnell finanziell zu Buche schlagen.

Literaturverzeichnis

[1] Beschreibung zu PAX. http://pax.grsecurity.net/.

[2] Homepage des Adamantix-Projekts. http://adamantix.org.

[3] Homepage des Sniffix-Projekts. http://sniffix.org.

[4] Homepage von FreeBSD. http://www.freebsd.org/.

[5] Homepage von RSBAC. http://www.rsbac.org.

[6] Homepage von SELinux der NSA. http://www.nsa.gov/selinux/.

[7] Homepage von TrustedBSD. http://www.trustedbsd.org/.

[8] W. Salamon S. Smalley, C. Vance. Implementing SELinux as a Linux Security Module. http://www.nsa.gov/selinux/module-abs.html.

[9] S. Smalley. Configuring the SELinux Policy. http://www.nsa.gov/selinux/policy2-abs.html.

6 CORBA: Mechanismen der Durchsetzung und Verwaltung von Rechten und Politiken

Eldar Sultanow, Alexander Renneberg

Abstract

Die Ausarbeitung gibt am Anfang einen detailierten Überblick über die CORBA Architektur und wichtigsten Fachbegriffe. Dazu zählen die Interface Definition Language, Inter-ORB Protocol und Object Services.
Es werden die CORBASecurity Fachbegriffe privilegs, principal und credentials und deren Einfluss auf die Sicherheitsrichtlinien wie Delegation, Unabstreitbarkeit und Access Control herausgearbeitet.
CORBA und CORBASecurity sind eine Spezifikationen, die von unterschiedlichen Software-Herstellern implementiert werden. Daraus ergeben sich aufgrund fehlender Kompatibilität und Absprachen Probleme für Sicherheit, denn diese kann verantwortungsvoll nur auf Anwendungsebene realisiert werden. Außerdem werden große Teile wie z.B. ein spezifizierter Audit Service von keinem Hersteller ordentlich implementiert.

6.1 Einleitung

Große verteilte Anwendungen spielen in der heutigen Informationsgesellschaft eine immer größere Rolle. Durch die Vielfältigkeit und Offenheit dieser Systeme ergibt die sich Frage nach Sicherheit.
Die große technologische Evolution in den letzten Jahren führte zu der Situation, dass immer größer werdende, verteilte Anwendungen, die miteinander kommunizieren, einen immer wichtigeren Teil der IT Technologie darstellen. Als Konsequenz wurden Standards, Architekturen und Frameworks zur Entwicklung dieser komplexen Anwendungen geschaffen. Die Object Management Group (OMG) spezifizierte daraufhin die Object Management Architecture (OMA). Im Herzen dieser OMA befindet sich die CORBA Spezifikation.
Die CORBA Spezifikation ermöglicht es Entwicklern, verteilte Anwendungen nach dem objekt-orientierten Paradigmen zu entwerfen und zu implementieren. Dieses geschieht nach einem standardisierten Muster, welches die Portabilität und Interoperabilität sichert. Als zentrale Komponente in der CORBA Architektur fungiert der Object Request Broker (ORB).

Die CORBA Security Service behandelt nur Schnittstellen, die von jedem einzelnen ORB implementiert werden müssen.
Policies können sowohl vom ORB, als auch von der Anwendung umgesetzt und durchgesetzt werden.
Die Sicherheit Funktionalität, die durch die CORBA Security Spezifikation definiert wird, enthält folgende Punkte.

- Die Kennzeichnung und Authentifizierung von Principals (menschliche Nutzer und Gegenstände, die ihre eigenen Recht haben müssen) sind dazu da, um zu überprüfen, welche Identität hinter einem Principal steht.
- Authorisierung und Infrastruktur-basierte Zugriffskontrolle entscheiden, ob ein Principal auf eine Object Domain, ein einzelnes Object oder eine Operation auf ein Object zugreifen kann, normalerweise mit den Identität und/oder Privilegattributen des Principals (wie Rolle, Gruppen).
- Security Auditing zeichnet in unrevidierbarer Weise auf, welcher Benutzer etwas Sicherheitsrelevantes gemacht haben. Es ist normalerweise der menschliche Benutzer, der Verantwortlichkeiten tragen sollte. Audit Mechanismen müssen in der Lage sein, den Benutzer, sogar nach einer Kette von Anrufen, richtig zu identifizieren.

Die Sicherheit der Kommunikation zwischen Gegenständen findet häufig über unsichere Kommunikationsschichten ab. Dieses erfordert Vertrauen, welches zwischen dem Client und dem Ziel hergestellt werden soll, welches wiederum die Authentifizierung der Klienten zu dem Server einschließt. Auch wenn diese Authentifizierung vollständigen Integritäts- und Vertraulichkeitsschutz bei dem Transfer der Datenpakete voraussetzt.

6.2 CORBA im Überblick

Zwei eindeutige Schnittstellen können innerhalb dieser Architektur unterschieden werden. Die horizontale Schnittstelle wird zwischen der Anwendung und dem ORB definiert. Mit Hilfe dieser Schnittstelle interagiert die Anwendung mit der unterliegenden CORBA Infrastruktur. Serverseitig implementiert die Anwendung Objekte, welche dem Client innerhalb eines Netzwerkes bestimmte Dienste anbietet. Diese Dienste (das Angebot an Services) werden in einer programmiersprachenunabhängige Weise mit der CORBA Interface Definition Language (IDL) spezifiziert. Client-seitig kann die Anwendung diese Dienste vom Server wiederum durch einen Zugriff auf ein anwendungsspezifisches Stub nutzen.
Jede beliebige Programmiersprache kann an dieser Stelle für die Implementierung zum Einsatz kommen, soweit eine Abbildung der Sprache auf eine IDL existiert. Zudem eignet sich jede Plattform (Betriebssystem), für welches sich ein ORB System auf dem Markt befindet. Auf diese Art und Weise erreicht CORBA die Portabilität von Anwendungscode und Wiederverwendung von Legacy- Code.
Die zweite Schnittstelle stellt die vertikale Schnittstelle dar. Diese ist die Kommunikationsschnittstelle eines ORB Systems, welche auf unterschiedlichen Knoten

innerhalb eines Netzwerkes agiert. An dieser Stelle wird von jedem ORB kompatiblen Produkt ein standardisiertes Protokoll verwendet. Für TCP/IP basierte Netze ist dieses das Internet Inter-ORB Protokoll (IIOP). Dieses Protokoll garantiert auf diese Art und Weise die Interoperabilität zwischen Installationen von CORBA Produkten von verschiedenen Herstellern. Diese gehören zu unterschiedlichen technischen Domains.
Um es einfach zusammenzufassen, kann der ORB als ein Software Bus beschrieben werden, welches die Kommunikation zwischen Client und Server sicherstellt.
Die ORB Spezifikation repräsentiert die Backbone der CORBA Architektur und bietet ein Framework an, welches Portabilität, Wiederverwendbarkeit und Interoperabilität anbietet. Nichtsdestotrotz stellt CORBA nur unzureichend viel Unterstützung für Entwickler von verteilten Anwendungen zur Verfügung.
Wenn diese Entwickler eine verteilte Anwendung realisieren, müssen diese sich mit verschiedenen Problemen, wie das Auffinden von Diensten, das Garantieren von transaktioneller Abarbeitung oder das Unterstützen von asynchroner Kommunikation, auseinandersetzen. Diese Probleme werden nämlich nicht innerhalb des ORBs gelöst, denn es wurde bei dem ORB nur auf die Kernfunktionalität Wert gelegt. Allerdings spezifiziert die OMG aus diesem Grund Object Services, die sich außerhalb des ORB Kerns befinden. Beispiele dafür sind der Naming Service, der Trading Service, der Transaction Service, den Event Service und am Ende den Security Service (CORBAsec).

6.3 Überblick - CORBA Security

Im letzten Abschnitt wurde beschrieben, dass die CORBAsec das Ziel verfolgt, die Sicherheit innerhalb der ORB Umwelt in Form von Object Services zur Verfügung zu stellen. Der Schwerpunkt liegt hierbei auf Vertraulichkeit (confidentiality), Integrität und Verantwortlichkeit (accountability). Technisch gesehen werden diese Dienste durch die OMG Spezifikation und die Implementierung (von einem CORBAsec Produkthersteller) von Objekten realisiert. Dabei decken diese Dienste eine Vielzahl von Schnittstellen ab, welche in der CORBA IDL definiert wurden.
Eine Beschreibung dieser Schnittstellen/Funktionalitäten mit den tiefer-liegenden Entwurfsrichtlinien findet sich im kommenden Abschnitt.
Terminologie und Hauptkomponenten:
In der CORBAsec werden die Nutzer des Systems (actors) mit dem Begriff principals bezeichnet. Jeder principal wird mit einem credential assoziiert, welches die Sicherheitsattribute (privileges) zusammenfasst.

6.4 Verwaltung von Rechten und Politiken

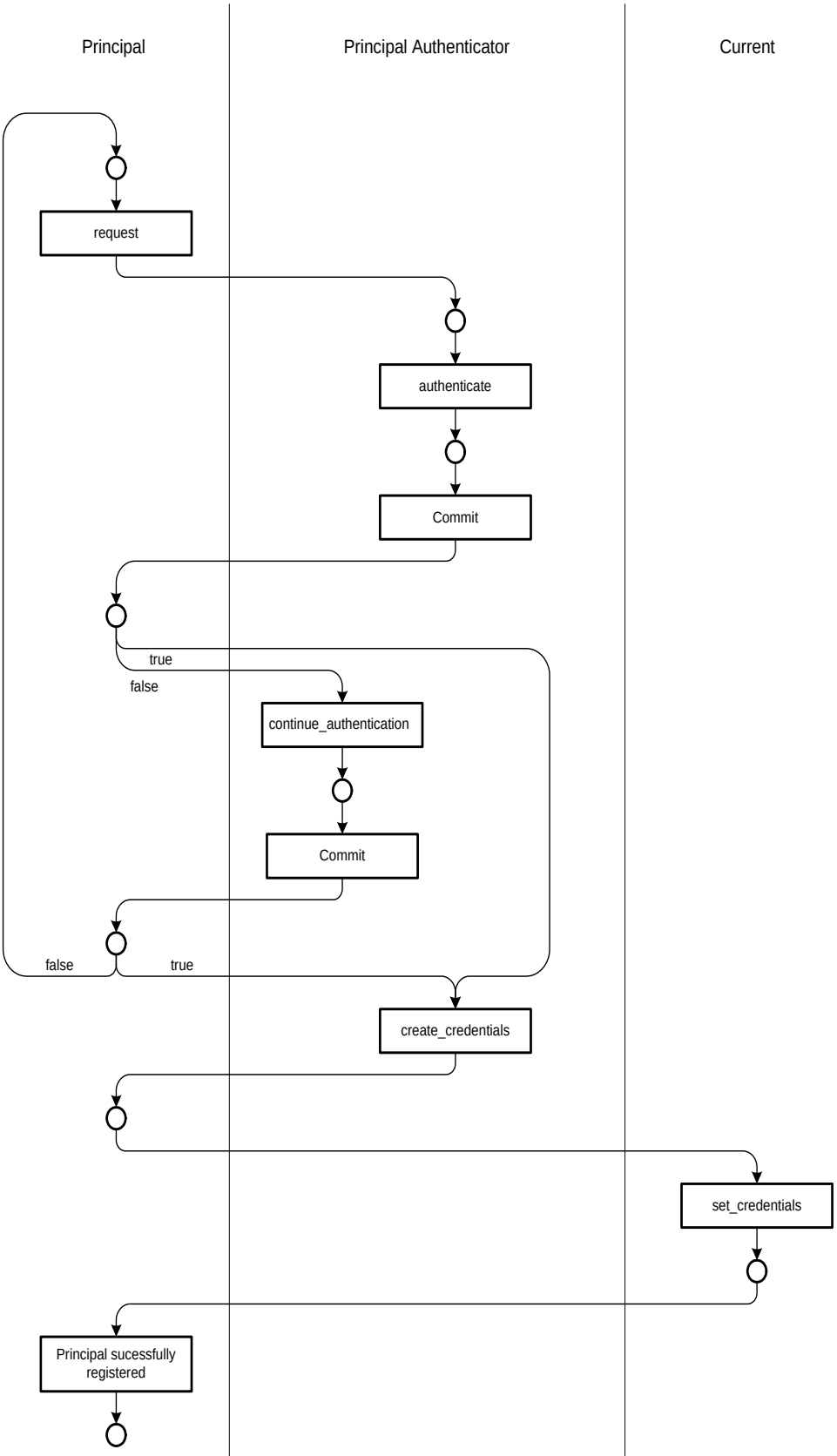
6.4.1 Authentifizierung

Der erste Schritt hin zu einer sicheren CORBA-Anwendung ist die Authentifizierung der Objekte/Benutzer. Hierdurch weist ein auf das System zugreifender Principal eindeutig seine Identität nach. Bei einem Principal kann es sich sowohl um ein CORBA-Objekt, als auch um einen menschlichen Benutzer handeln. Ein

menschlicher Benutzer wird gegenüber der CORBA- Anwendung durch einen User Sponsor vertreten.
Die Authentifizierung läuft unabhängig von der verwendeten Sicherheitstechnologie folgendermaßen ab (siehe Abbildung 1):
Im ersten Schritt ruft der Principal die Methode authenticate im Interface PrincipalAuthenticator auf. Dieser Methode wird die gewünschte Authentifizierungsmethode und die dazu benötigten Daten übergeben. Als Ergebnis gibt diese Methode zurück, ob die Authentifizierung erfolgreich war oder ob ein weiterer Authentifizierungsschritt nötig ist.
In diesem Fall wird die Methode continue_authentication aufgerufen. Verläuft die Authentifizierung erfolgreich, so erzeugt der PrincipalAuthenticator ein Credentials-Objekt für den Principal. Eine Referenz auf dieses Objekt ist ein Rückgabewert der Authentifizierungsmethode. Mit der Methode set_credentials in der Schnittstelle Current ordnet der User Sponsor dieses Credentials-Objekt dem aktuellen Ausführungskontext zu. Bei einem zukünftigen Methodenaufruf wird dieses Credentials-Objekt verwendet.
Das Clientobjekt ist nun erfolgreich am System angemeldet. Weitere Anfragen an Serverobjekte erfolgen nun auf herkömmliche Art und Weise, ohne dass der Client etwas von den Sicherheitsmaßnahmen bemerkt. Wird Level-1-Sicherheit eingesetzt, so weiß ein Client nichts von Sicherheit und enthält auch keinen Code, der obige Authentifizierung durchführen kann. In diesem Fall ist es möglich, die Authentifizierung für ein Objekt extern durchzuführen. So wird zum Beispiel die Authentifizierung mit Kommandozeilenparametern des ORB durchgeführt. Beim Start des Objektes sind dann schon die entsprechenden Credentials dem Ausführungskontext des Objekts zugeordnet.
Abbildung 1. Authentifizierung im CORBA-Security-Service

6.4.2 Credentials und Delegation

Wie eben erläutert, erhält ein Objekt nach einer erfolgreichen Authentifizierung ein Credential-Objekt. Ein Credential ist der Ausweis eines Objektes im System, der für alle weiteren Sicherheitsüberprüfungen verwendet wird (Single Sign On). In diesem Objekt sind die Sicherheitsattribute eines Principals gespeichert, u.a. Rechte und authentifizierte Identitäten. Auch anonyme Benutzer des Systems besitzen ein Credential-Objekt, allerdings mit geringeren Rechten.
Der genaue Inhalt des Credential-Objekts hängt wieder vom konkreten Sicherheitsmechanismus ab. Der Zugriff erfolgt über die Schnittstelle Credentials. Es gibt allerdings keine Möglichkeit, die im Objekt hinterlegten sicherheitsrelevanten Daten auszulesen. Über die zwei Methoden set_security_features und set_privileges kann festgelegt werden, welche Sicherheitsattribute und –funktionalitäten bei einem bestimmten Aufruf eingesetzt werden. Über die Funktion copy ist es möglich, eine Kopie eines Credential-Objektes anzufertigen. Dadurch ist es möglich, für verschiedene Zielobjekte angepasste Credential-Objekte zu verwenden. In einem objektorientierten System kann die Anfrage eines Clients nicht allein vom Server bearbeitet werden.
Das Zielobjekt der Anfrage muss oft wiederum andere Objekte mit Teilaufgaben beauftragen. Dies kann sich fortsetzen, so dass sich schließlich ein Baum aus An-


Principal
Principal Authenticator
Current
request
authenticate
Commit
true
false
continue_authentication
Commit
false
true
create_credentials
set_credentials
Principal sucessfully registered

fragen aufbaut, dessen Wurzel der Initiator der Anfrage ist. Hierbei ist es möglich, dass zwischenliegende Objekte Aktionen im Namen des Initiators durchführen müssen. Es ist also nötig, dass die Sicherheitsattribute des Initiators mit den Methodenaufrufen weitergeleitet werden (Delegation). Im CORBA-Security-Service sind fünf verschiedene Delegationsmechanismen spezifiziert:

- No Delegation
- Simple Delegation
- Composite Delegation
- Combined Privileges
- Delegation
- Traced Delegation

Bei ,,No Delegation“ können die Sicherheitsattribute vom zwischenliegenden Objekt nur zum
Zweck der Zugriffskontrolle eingesetzt werden. Das zwischenliegende Objekt handelt bei weiteren Methodenaufrufen in seinem eigenen Namen. Bei ,,Simple Delegation“ gibt das zwischenliegende Objekt einfach das Credential-Objekt des Clients weiter. Das Zielobjekt kann nicht feststellen, wer das Zwischenobjekt ist.
Es stellt eine Verkörperung (impersonation) des Clients dar. Bei der ,,Composite Delegation“ und ,,Combined Privileges Delegation“ hat das Zwischenobjekt das Recht, die Sicherheitsattribute des Clients zu verwenden und weiter zu delegieren. Der Unterschied zwischen den beiden Delegationsarten ist, dass bei der ,,Composite Delegation“ die Credentials getrennt weitergegeben werden.
Das Zielobjekt kann also beide Credentials getrennt überprüfen und kennt somit sowohl den Initiator als auch das Zwischen-objekt. Bei der ,,Combined Delegation“ werden die Attribute der beiden Objekte gemeinsam weitergegeben, so dass das Zielobjekt nicht nachvollziehen kann, wer welche Rechte besitzt. Bei der ,,Traced Delegation“ werden in jedem Zwischenobjekt die eigenen Credentials angefügt, so dass das Zielobjekt eine Kette von Credentials aller Zwischenobjekte erhält. Aus Sicht des Client sehen die letzten drei Delegationsarten gleich aus. Zusätzlich zu diesen Möglichkeiten können Implementierungen für die Delegation noch Zeitbeschränkungen und eine Beschränkung von Methodenaufrufen anbieten.

6.4.3 Zugriffskontrolle

Die Abbildung 2 zeigt die Architektur der Zugriffskontrolle im CORBA-Security-Service. Er besteht aus zwei Ebenen: Die untere Ebene beinhaltet die Zugriffskontrollmechanismen, die automatisch von einem sicheren ORB durchgeführt werden (Invocation Access Decision). Diese Zugriffskontrolle kann für den Client und den Server vollkommen transparent erfolgen und ist somit in beiden Sicherheits-Leveln verfügbar.
Die obere Ebene enthält die Zugriffskontrollmechanismen mit denen die Anwendung selber ihre Sicherheitsanforderungen durchsetzen kann. In diesem Modell

wird zwischen einer Zugriffskontrolle auf Client bzw. auf Server-Seite unterschieden. Auf Client-Seite wird kontrolliert, ob der Client eine Methode des Servers aufrufen darf. Auf Server-Seite wird entschieden, ob unter den gegebenen Umständen der Server den Aufruf annehmen muss. In einer realen Implementierung kann unter Umständen die Prüfung auf einer der beiden Seiten weggelassen werden.
Die eigentliche Entscheidung, ob nun ein Zugriff stattfinden darf, wird von einem Objekt des Typs AccessDecision durchgeführt. In diesem Interface ist hierzu die Methode access_allowed definiert, der eine Liste aller Credentials, das Zielobjekt, der Name der aufgerufenen Methode und der Name der Schnittstelle des Zielobjektes übergeben werden. Die Entscheidung kann getroffen werden anhand der Identität des Aufrufers, anhand einer Rolle oder der Zugehörigkeit zu einer Gruppe oder anhand der Rechte, die ein Aufrufer besitzt. Darüber hinaus können auch Parameter des Methodenaufrufes in diese Entscheidung mit einfließen. So sei es einem bestimmten Benutzer erlaubt, sein Konto um 10.000 DM zu überziehen. Beim Aufruf einer Methode, die Geld vom Konto abbucht, muss also bei jedem Zugriff kontrolliert werden, ob der Kreditrahmen eingehalten wird.
Diese Art der Zugriffskontrolle kann auf der Ebene des ORB nicht automatisch durchgeführt werden, da hierzu anwendungsbezogenes Wissen nötig ist. Solche Sicherheitsanforderungen müssen auf Anwendungsebene durchgesetzt werden und sind somit nur bei Anwendungen möglich, die Level-2-Sicherheit verwenden.
TRIAL RESTRICTION
Abbildung 2. Zugriffskontrolle im CORBA-Security-Service
Die vom CORBA-Security-Service automatisch durchgeführte Standard-Zugriffskontrolle erfolgt über Rechte. Implementierungen des Dienstes können jedoch noch zusätzlich eigene Mechanismen einsetzen, die nicht auf Rechten basieren.
Für jeden Methodenaufruf ist also eine definierte Menge von Rechten nötig, die für das aufgerufene Objekt festgelegt wurden. Über die Schnittstelle RequiredRights wird festgelegt, welche Rechte zum Aufruf einer Methode einer bestimmten Schnittstelle nötig sind. Hierbei kann angegeben werden, ob alle übergebenen Rechte nötig sind oder nur eines davon.
Über die Schnittstelle DomainAccessPolicy werden bestimmte Rechte bestimmten Identitäten im System zugeordnet. Es wird also festgelegt, welcher Nutzer innerhalb des Gültigkeitsbereichs dieser Zugriffspolicy über welche Rechte verfügt. Die Rechte werden in sogenannten Rechte-Familien (rights family) zusammengefasst. Im Standard ist die Familie ,,CORBA“ definiert, die folgende Rechte enthält:

- g für get
- s für set
- m für manage
- u für use

Reichen diese Rechte nicht aus, um die Anforderungen einer Anwendung auszudrücken, so können neue Familien mit beliebigen Rechten angelegt werden. So kann z.B. für eine Bank- Applikation eine neue Familie ,,Bank“ mit den Rechten ,,call balance“ und ,,create account“ erstellt werden. Das erste Recht wird allen

Bankkunden erteilt, das zweite Recht haben nur Bankangestellte. Bei der automatischen Zugriffskontrolle im CORBA-Security-Service ist zu beachten, dass alle Mechanismen auf Schnittstellen-Ebene arbeiten.
Man kann festlegen, wie der Zugriff auf alle Objekte, die eine bestimmte Schnittstelle implementieren, geregelt ist. Es ist allerdings nicht möglich, den Zugriff auf einzelne Instanzen einer Schnittstelle zu regeln. Bei obigem Bank-Beispiel könnten also alle Bankkunden auch den Kontostand von fremden Konten einsehen.
Ein Ausweg aus dieser Problematik wäre für jedes Objekt eine eigene Policy zu definieren, was aber zu einem riesigen Administrationsaufwand führen würde. Soll also der Zugriff auf einzelne Objektinstanzen geregelt werden, so bleibt einem nichts anderes übrig, als auf Applikationsebene seine eigenen Sicherheitsanforderungen durchzusetzen.

6.4.4 Auditing

Unter Auditing versteht man die Aufzeichnung von sicherheitsrelevanten Ereignissen, um tatsächliche oder versuchte Verstöße gegen die Sicherheitsbestimmungen zu bemerken und Maßnahmen dagegen ergreifen zu können. Beispiele für solche interessanten Ereignisse sind z.B. die Authentifizierung von neuen Nutzern, der Erfolg bzw. Misserfolg von Methodenaufrufen oder die Administration von Nutzern und Sicherheitspolicies.
Abhängig von einer bestimmten Anwendung kann es weitere Ereignisse geben, die aufgezeichnet werden sollten. Ähnlich wie bei der Zugriffskontrolle wird auch beim Auditing zwischen der ORB-Ebene und der Anwendungsebene und auf jeder Ebene zwischen Client- und Serverseite unterschieden. Auf der ORB-Ebene wird die Aufzeichnung von relevanten Systemereignissen durchgeführt.
Dies geschieht automatisch ohne weiteres Zutun der Anwendung und ist somit auch bei Anwendungen, die Level-1-Sicherheit verwenden, verfügbar. Die Entscheidung, ob ein Ereignis gespeichert werden soll, wird von einem Objekt des Typs Audit_Decision mit der Methode audit_needed getroffen. Zu dieser Entscheidung können folgende Kriterien herangezogen werden:

- der Objekttyp
- der Name der aufgerufenen Methode
- die Zeit
- die Sicherheitsattribute des Aufrufers
- der Erfolg/Misserfolg des Methodenaufrufs

Fällt die Entscheidung für das Auditing aus, so wird ein Audit-Datensatz in einen sogenannten Audit-Channel mit der Methode audit_write geschrieben. Im Standard ist nur diese Schnittstelle spezifiziert. Wie der Datensatz anschließend sicher gespeichert und wie die Auswertung der gesammelten Daten erfolgen soll, darüber schweigt sich die Spezifikation aus.
Falls also hier spezielle Anforderungen hinsichtlich der sicheren Speicherung von Auditingdaten existieren, muss dieser Punkt bei der einzelnen Implementierung

geprüft werden. Zusätzlich zu den Ereignissen, die automatisch vom ORB aufgezeichnet werden, kann es abhängig von der speziellen Anwendung noch zusätzliche sicherheitsrelevante Vorkommnisse geben.
So möchte man z.B. in einer Bank-Anwendung Auszahlungen von über 5.000 Euro aufzeichnen. Da hierfür inhaltliches Wissen über die Anwendung notwendig ist, kann dies nicht vom ORB automatisch erledigt werden, sondern die Anwendung muss selber einen Auditing-Datensatz erstellen und ihn in einen Audit-Channel schreiben.

6.4.5 Unabstreitbarkeit

Die Unterstützung für Non-Repudiation (Unabstreitbarkeit) handelt es sich um eine optionale Funktionalität. Mit diesem Dienst ist es möglich, einen unwiderruflichen Beweis für eine erfolgte Aktion zu erstellen und diesen für den Fall einer späteren Meinungsverschiedenheit sicher zu speichern. Hierdurch ist es nicht mehr möglich, im Nachhinein etwas abzustreiten, was dazu führt, dass jeder Nutzer für seine Taten verantwortlich gemacht werden kann.
Bei Methodenaufrufen werden zwei Arten von Beweisen erstellt. Damit ein Initiator eines Methodenaufrufes diesen nicht abstreiten kann, wird ein Herkunftsbeweis (proof of origin) erstellt. Dies geschieht indem der Client die Methode generate_token im Interface NRCredentials aufruft (1). Bei einem Objekt vom Typ NRCredentials handelt es sich um eine spezielle Art von Credentials, die vom Non-Repudiation Service verwendet werden. Dieser Beweis wird mit dem Methodenaufruf zum Zielobjekt übertragen, das ihn mit der Methode verify_evidence in seinem NRCredentials-Objekt überprüft (2).
Anschließend wird vom Zielobjekt die Empfangsbestätigung (proof of receipt) erstellt (3). Dazu wird ebenfalls die Methode generate_token benutzt. Mit der Antwort des Methodenaufrufes wird dieser Beweis wiederum zum Client übertragen, der ihn mit Hilfe der Methode verify_evidence überprüfen kann (4).
Der ORB hat bei diesem ganzen Vorgang dafür zu sorgen, dass die Beweise geschützt zwischen Client und Server übertragen werden und dass kein Beweis mehrmals verwendet werden kann.
Erreicht wird dies normalerweise durch den Einsatz von digitalen Signaturen und von Zeitstempeln. Im Standard ist jedoch keine spezielle Technologie vorgeschrieben. Da die Beweise von der Client- bzw. von der Server-Anwendung erzeugt werden müssen, steht die Funktionalität des Non-Repudiation Service nur bei Level-2-Sicherheit zur Verfügung.
TRIAL RESTRICTION
Abbildung 3. Unabstreitbarkeit im CORBA-Security-Service
Die untere Hälfte von Abbildung 3 zeigt das ISO Non-Repudiation Modell, auf dem die Unterstützung für Unabstreitbarkeit im CORBA-Security-Service beruht. Die einzige Komponente
dieses Modells, die vom CORBA-Security-Service unterstützt wird, ist, wie oben beschrieben, die Beweiserstellung und die Beweisüberprüfung. Auf die Überbringungsinstanz (Delivery Authority), die für Überbringung von Nachrichten und den zugehörigen Beweisen entsprechend der zuständigen Non-Repudiation Policy verantwortlich ist, wird im Standard nicht eingegangen. Die Beweisspeicherung, die für eine sichere

persistente Speicherung der Beweise zuständig ist, wird nicht spezifiziert, da hierfür andere CORBA-Dienste, wie z.B. der Persistent State Service, verwendet werden können. Ebenfalls unspezifiert ist ein Schiedsrichter-Objekt (Adjudiactor), das im Falle eines Streites eine Entscheidung fällt.
Die Tatsache, dass wichtige Aspekte des Non-Repudiation-Dienstes nicht im Standard festgelegt sind, führt dazu, dass jeder Hersteller hier eine eigene Lösung wählen kann. Diese ungenaue Spezifikation ist eventuell mit ein Grund dafür, dass diese optionale Funktionalität bis jetzt in keiner Implementierung des Security Service zur Verfügung steht.

6.4.6 Administration und Policies

Wenn jedes einzelne Objekt in einem verteilten System individuell verwaltet werden muss, so
wächst der Administrationsaufwand mit der Größe des Systems. Um dies zu verhindern, können Objekte, die gleiche Sicherheitsanforderungen haben, zu Domänen (domains) zusammengefasst werden. Zusätzlich zu den gemeinsamen Sicherheitsanforderungen besitzen alle Objekte einer Domäne, eine gemeinsame Stelle, die für ihre Verwaltung zuständig ist. Die Sicherheitsanforderungen einer Domäne werden mit Policies (domain security policies) für verschiedene Aspekte festgelegt.
Durch den ORB werden diese Anforderungen für alle Objekte, die Mitglied in einer bestimmten Domäne sind, automatisch durchgesetzt. Durch diese Gruppierung von Objekten können auch sehr große verteilte Systeme effizient verwaltet werden. Eine Unterdomäne besitzt zunächst dieselben Policies wie die Oberdomäne.
Diese können aber bei Bedarf angepasst werden. Wird nun eine Methode eines Domänen-Objektes aufgerufen, so werden automatisch vom ORB die entsprechenden Policies angewendet. Für die folgenden Bereiche können Policies definiert werden:

- Zugriffskontrolle: Mit einer Access Policy wird definiert, welcher Principal in dieser Domäne über welche Rechte verfügt.
- Auditing: Mit der Audit Policy wird festgelegt, welche sicherheitsrelevanten Ereignisse im System aufgezeichnet und welche Daten jeweils gespeichert werden sollen.
- Schutz der Kommunikation: Mit der Secure Invocation Policy wird festgelegt, wie die Übertragung der Nachrichten zwischen den einzelnen Objekte geschützt werden soll. Hierbei existieren die folgenden Optionen:
  - Kein Schutz
  - Schutz hinsichtlich Integrität
  - Geheimhaltung der Übertragung
  - Entdeckung von Replay-Attacken
  - Erhaltung der Reihenfolgetreue bei mehrteiligen Nachrichten
  - Authentifizierung von Client bzw. Server

– Delegation: Mit der Delegation Policy wird die Art der Delegation spezifiziert. Es kann zwischen ,,No Delegation“, ,,Simple Delegation“ und ,,Composite Delegation“ gewählt werden.
– Non-Repudiation: Mit der Non-Repudiation Policy werden Regeln für die Erstellung von Beweisen bzw. die Überprüfung von Beweisen definiert.

Die Schnittstellen, über die auf die oben genannten Policy-Objekte zugegriffen werden kann,
sind alle im Modul SecurityAdmin spezifiziert. Mit den spezifizierten Methoden können
allerdings nur die Eigenschaften der Policy-Objekte, nicht aber die Zugehörigkeit von Objekten
zu einer Domäne, beeinflusst werden.

6.4.7 Interceptoren

Ein Interceptor ist eine Art Unterbrechung der normalen Bearbeitung eines Aufrufes im ORB. Innerhalb dieser Interceptor-Funktionen können zusätzliche Maßnahmen, z.B. für die Absicherung, ergriffen werden. Es werden zwei Arten von Interceptoren unterschieden:

- Request-level Interceptors. Die Methoden dieses Interceptoren werden pro Aufruf einmal aufgerufen. Ein Beispiel für einen Requestlevel Interceptor ist der Access Control Interceptor, über den die Zugriffskontrolle abgewickelt wird. Die Methoden, die im Interface CORBA::RequestInterceptor definiert sind, heißen client_invoke auf Client-Seite, bzw. target_invoke auf Server-Seite. Als Parameter wird den Methoden ein strukturiertes Request-Objekt übergeben.

- Message-level Interceptors. Dieser Interceptor wird pro Nachricht, die über das darunterliegende Netzwerk übertragen wird, aufgerufen. Dies kann mehrmals pro Methodenaufruf der Fall sein. Beim Empfang der Nachricht wird die Methode receive_message in der Schnittstelle CORBA::MessageInterceptor aufgerufen, zum Senden die Methode send_message. Diese Methoden arbeiten auf einem unstrukturierten Byte-Puffer.

Interceptoren befinden sich zwischen Client und Server und operieren auf Schicht der IIOP Nachrichten. Für Server existieren zwei Arten der Interceptoren: zum einen pro Verbindung, zum anderen pro Prozess. Interceptoren auf Client Seite können pro Objekt oder ebenfalls pro Prozess existieren.
Für den Server gilt, dass im Fall eines Interceptors pro Verbindung für jede Verbindung zwischen einem Client und dem Server eine neue Instanz des Interceptors erzeugt wird. Bei einem Interceptor pro Prozess ist der Interceptor für den gesamten Nachrichtenverkehr zwischen den Clients und dem Server zuständig. Im Fall eines Interceptors pro Objekt wird für jedes individuelle Serverobjekt ein Interceptor instanziiert.
Zusätzlich existiert ein Interceptor-Typ, der als BindInterceptor bezeichnet wird und dessen Methoden während des Bindevorgangs eines Clients an ein Objekt

aufgerufen werden. Es können mehrere Interceptoren gleichzeitig installiert werden. Diese bilden eine Kette. Die IIOP Nachrichten werden dann nacheinander an die installierten Interceptoren weitergereicht. Die Installation erfolgt in den ORB und muss erfolgen, bevor die Anwendungsobjekte
instanziiert werden. In der ORB-Implementierung von VisiBroker existieren drei Java-Interfaces die implementiert werden können um Interceptor-Funktionalität bereitzustellen.

- ServerInterceptor: Er deklariert Methoden, die direkt nach Erhalt einer Anfrage oder vor dem Absenden einer Antwort aufgerufen werden. Es ist möglich auf die übermittelten Daten zuzugreifen und diese zu verändern.
- ClientInterceptor: Er deklariert Methoden, die unmittelbar vor dem Absenden einer Anfrage oder dem Erhalt einer Antwort aufgerufen werden. Es ist möglich auf die übermittelten Daten zu-zugreifen und diese zu verändern.
- BindInterceptor: Er deklariert Methoden die aufgerufen werden, wenn ein Client einen bind()- oder rebind()-Aufruf macht.

Die CORBA Security Service behandelt nur Schnittstellen, die von jedem einzelnen ORB implementiert werden müssen.
Policies können sowohl vom ORB, als auch von der Anwendung umgesetzt und durchgesetzt werden.
Die Sicherheit Funktionalität, die durch diese Spezifikation definiert wird, enthält:
Kennzeichnung und Authentifizierung von Principals (menschliche Nutzer und Gegenstände, die ihre eigenen Recht inne haben müssen) sind dazu da, um zu überprüfen, welche Identität hinter einem Principal steht.
Authorisierung und Infrastruktur-basierte Zugriffskontrolle - entscheiden, ob ein Principal auf eine Object Domain, ein einzelnes Object oder eine Operation auf ein Object zugreifen kann, normalerweise mit den Identität und/oder Privilegattributen des Principals (wie Rolle, Gruppen).
Security Auditing zeichnet in unrevidierbarer Weise auf, welcher Benutzer etwas Sicherheitsrelevantes gemacht hat. . Es ist normalerweise der menschliche Benutzer, der Verantwortlichkeiten tragen sollte. Audit Mechanismen müssen in der Lage sein, den Benutzer, sogar nach einer Kette von Anrufen, richtig zu identifizieren.
Die Sicherheit der Kommunikation zwischen Gegenständen findet häufig über unsichere Kommunikationsschichten ab. Dieses erfordert Vertrauen, welches zwischen dem Client und dem Ziel hergestellt werden soll, welches wiederum die Authentifizierung der Klienten zu dem Server. Auch wenn diese Authentifizierung vollständigen Integritäts und Vertraulichkeitsschutz bei dem Transfer der Datenpakete.
Non-Repudiation (Nicht-Ablehnung) stellt unwiderlegbare Beweise von Tätigkeiten wie Herkunftsnachweise von Daten vom Empfänger oder Beweise zum Empfang von Daten zum Absender zur Verfügung, um sich gegen folgende Versuche zu schützen:
Das Empfangen und Senden von Daten wird fälschlicherweise verweigert.
Das CORBA Sicherheit Modell ist ein sicherheitstechnikneutrales Framework.

Schnittstellen, die für die Sicherheit von client-target object invocation, kapseln die Sicherheitsmechanismen, die von den Anwendungsobjekten und dem ORB genutzt werden.
Es ist möglich, CORBA Sicherheit auf einer breiten Vielzahl der vorhandenen Systeme einzuführen. Damit werden die Sicherheitsprotokolle wieder verwendet, die schon nativ zu einem System gehören.
Der CORBA Sicherheit Service kann Zugang zu Anwendungs- Objekten realisieren, ohne dass das die Anwendung merkt. Damit kann der Service auf andere Systeme portiert werden, die andere Security Policies durchsetzen möchten.
Diese Spezifikation definiert den Kernsicherheit Service und die Schnittstellen, die erfordert werden, um ein angemessenes Niveau an Sicherheit eines CORBA-konformen System als Ganzes sicherzustellen.

6.5 Security Policies

CORBA Security Services bieten ein hohes Maß an Flexibilität an. Zum einen ändern sich die Anforderungen an Security Policies. Ein Unternehmen (Anwendungsfeld) sollte ein solches Maß an Sicherheit und Schutz anwenden, welches die Umwelt fordert. Folgende Typen von Flexibilität bei den Policies spielen eine Rolle:
Auswahl an Access Control Policies
Die definierte Schnittstelle bietet eine breite Auswahl von Mechanismen an. ACL (access control lists) nutzen eine Breite von Privilegien Attribute wie Identitäten, Rollen, Gruppen oder Labels. Die Details sind von der Anwendungsschicht gekapselt, mit Ausnahme von einigen administrativen Funktionalitäten.
Auswahl an Audit (Prüf) Policies
Die Event Types (Ereignistypen), die überwacht und geprüft werden sollen, sind konfigurierbar. Dieses macht es möglich, den Aufwand und die Größe des Prüfpfades und auch die damit einzusetzenden Ressourcen zu bestimmen.
Die Unterstützung für Security Functionality Profiles wird gewährleistet. Diese wurden von verschiedenen nationalen und internationalen Regierungsorganisationen definiert (TCSEC – US Trusted Computer Evaluation Criteria) (ITSEC – European Information Technology Evaluation Criteria).

6.5.1 CORBA Security Interoperabilitäts Packages

Eine ORB Implementierung, die verschiedene Replaceability Packages unterstützt wird auch als Security Ready bezeichnet. Dazu nutzt das ORB Interzeptoren, um die Aufrufe für Security Services über das Replaceability Interface umzuleiten. Die Interfaces sind so eingefügt, dass der Security Service nicht wissen muss, wie der ORB arbeitet (z.B. wo das geforderte Policy Object lokalisiert ist). Der ORB unterstützt nicht selbst die Sicherheitsfunktionalität, aber dieser ORB ist gerüstet diese Funktionalitäten des Replaceability Package zu nutzen (Hook Prinzip).
Das Package Common Secure Interoperability (CSI) Package unterstützt verschiedene Stufen (3 Stück) von Sicherheits-Interoperabilität. Diese Stufen können auf verteilten Systemen laufen, wo die Client und Server Objekte auf unterschiedlichen ORBs und Betriebssystemen laufen können. Ein ORB, das die CSI Stufe 2 unterstützt, ist vollständig mit der CORBA Security Specification konform.

Identitätsbasierte Policy ohne Delegation (CSI Level 0)
Auf dieser Stufe wird nur die Identität eines Principals ohne weitere Attribute vom Client zum Server übertragen. Wenn weitere Objekte aufgerufen werden, dann wird die Identität des Zwischenobjekts und nicht die des initiierenden Objektes angegeben. Delegation wird nicht unterstützt.
Identitätsbasierte Policy mit uneingeschränkter Delegation (CSI Level 1)
Auf dieser Stufe werden wiederum nur die Identitäten eines Principals übertragen, allerdings ist nun Delegation möglich. Bei weiteren Objektaufrufen wird die Identität des Initiators weitergeleitet. Es existieren allerdings keine Einschränkungen bei dieser Delegation, so dass Zwischenobjekte (intermediate objects) die initiierenden Objekte imitieren können.
Identitäts- und Privilegien basierte Policies mit gesteuerter Delegation (CSI Level 2)
Bei CSI Level 2 werden die Attribute des initiierenden Principals vom Client zum Ziel übertragen. Dazu kommen aber noch Daten zum Zugriffs und Prüf (Audit) Identitäten und eine Reihe von Privilegien (z.B. Rollen und Gruppen). Die Weiterleitung (Delegation) dieser Attribute zu anderen Objekten ist möglich, aber bestimmten Restriktionen ausgesetzt. Nur der initiierende Principal kann diese Restriktionen steuern. Zudem wird eine zusammenfassende Weiterleitung (composite delegation) unterstützt, so dass die Attribute von mehr als einem Principal übertragen werden können.
Die Security Policies von CSI Level 2 kann mit Hilfe des CSI-ECMA Protokoll für jedes ORB umgesetzt und durchgesetzt werden.
Dazu zählen die Identitäts- und privilegien-basierten Policies. CSI-ECMA (European Computer Manufacturers Association) kann mit Identitäten genutzt werden, aber der Administrator kann auch die Delegation Restriktionen deaktivieren, um damit auf CSI Level 1 zurückzufallen. Delegation kann auch ganz weggelassen werden, so der ORB auf Level 0 zurückfällt.
Für die benutzten Schlüssel (Keys) existieren zwei Varianten. Dazu zählen das Public/Private Key Konzept oder die Public Key Technologie für Schlüssel, die nur vertrauenswürdigen Instanzen verliehen werden.

6.5.2 Access Control Mode

Diese Abbildung verdeutlicht das einfache Framework für verschiedene Access Control Security Policies. Dieses Framework besteht aus zwei Schichten, den Object Invocation Access Policies, die automatisch mit jeder Object Invocation berücksichtigt werden. Dazu kommt noch die Application Access Policy Schicht, wo die Anwendung selbst die Policies durchsetzen kann.
Ein Client kann eine Operation auf einem Zielobjekt nur dann ausführen lassen, wenn dieses von der Object Invocation Policy erlaubt ist. Dieses wird durch Access Decision Functions durchgesetzt.
Die Funktionen basieren deren Entscheidungen auf verschiedenen Parametern.
Dazu gehören die aktuellen Privilegien-Attribute des Principal, jegliche Steuerungsattribute (z.B. wie lange ein Privileg gültig ist).

6.6 Probleme und Schwächen von CORBA Security

Die umfassenden Themen über die Schwachstellen von CORBA sind nicht nur technischer, sondern auch nicht-technischer Natur. Als erstes wird der Application Layer und die Architektur betrachtet.

6.6.1 Schwachpunkte auf dem Application Layer

CORBAsec wurde im Prinzip ursprünglich für statische Anwendungen in eingeschränkten Umwelten entwickelt und kann deshalb weniger einfach für neue Anforderungen und Trust Verbindungen für internetbasierte Anwendungen angepasst werden. Zum Beispiel ist die Codebasis von CORBAsec zu groß und meistens blocken Firewalls die Nachrichten zwischen den Objekten, so dass die verteilte Kommunikation dadurch gestört wird. Die OMG Firewall Spezifikation (Draft) liefert eine zunächst rudimentäre Integration von SSL in CORBA, dieses sind die ersten Versuche, um die CORBAsec an das Internet heranzuführen.
In der Theorie verfolgt die CORBAsec Architektur einen Schichtenansatz. Die Sicherheitsfunktionalitäten können entweder auf der Anwendungs- oder ORB Schicht eingefügt werden. Nichtsdestotrotz ist es sehr oft unklar, an welcher bestimmten Stelle eine Sicherheitsfunktionalität eingesetzt werden soll.
Wenn keine zusätzlichen Sicherheitsanforderungen implementiert werden müssen, dann empfiehlt es sich, Verschlüsselung außerhalb des CORBA Systems zu realisieren.
Eine weitere spezifische Schwäche der Architektur bleiben die covert channels, welche (z.B. object references IOR) sicherheitskritische Daten (wie Keys oder Namen von internen Servern) beinhalten. Außerdem ist es möglich ein Netzwerk nach unbekannten Objekten mit Hilfe der ,,access denied“ exception zu suchen, damit wird die Existenz von Servern aufgedeckt.

6.6.2 Authentifizierung / Authorisierung

Bevor ein Client und Server miteinander sichere Nachrichten umherschicken können, müssen diese die Identität und andere Sicherheitsattribute des Kommunikationspartners kennen. Die vordefinierten Attribute der CORBAsec sind sehr eingeschränkt und können nicht alle Eigenschaften (Properties) eines principals beschreiben, zudem bieten viele Sicherheitsmechanismen keine ausreichenden Sicherheitsattribute.
Das Authorisierungsmodell von CORBAsec weist entscheidende Schwachpunkte auf, wie z.B. die vordefinierten Zugriffsrechte (access rights), die nur eine beschränkte Berechtigungsdauer aufweisen und nicht in alle Anwendungsszenarien hineinpassen. Viele CORBA Produkthersteller führen zum Teil auch eigene Zugriffskontrollmodelle ein, die herstellerübergreifend nicht kompatibel sind. Aus diesem Grund ist es der einzige verantwortungsvolle Weg, Zugriffskontrollen einzuführen, um diese innerhalb der Anwendung zu integrieren.
Einige der aufgezeigten Probleme und Konflikte sind nicht immer direkt mit CORBAsec in Verbindung zu bringen, sondern beziehen sich konsequenterweise auf geerbte Schwierigkeiten in verteilten, objektorientierten Systemen. Dazu gehören die versus delegation oder die Zugriffskontroll versus inheritance.

6.6.3 Security Audits

Ein Security Audit ist ein Mechanismus eines sicheren Systems, welcher Informationen über sicherheitsrelevante Events in einer Log Datei aufzeichnet. Der Audit Mechanismus unterstützt die unabhängige Bewertung der Adäquatheit von Systemkontrollen, um die Kompatibilität zu etablierten Richtlinien (policy) zu testen und um Lücken in der Sicherheit aufzudecken.
Der Audit Service der CORBAsec kann ein vernünftiger Audit Mechanismus bieten, wenn die folgenden Schwächen adressiert und eliminiert werden. Zunächst ist die Spezifikation der Audit Funktionalität nicht vollständig, wie z.B. das Fehlen von Sicherheitsmechanismen von Audit Log-Dateien, effektive Filter Mechanismen oder Schnittstellen für die Analyse der Log-Dateien. Zudem existiert kein standardisiertes Format für die Audit Log-Dateien, es ist somit unmöglich, eine zentrale Audit Abarbeitung zu implementieren. Für die Portabilität sollten nur Attribute des dazugehörenden CSI Level genutzt werden, allerdings beeinflusst dieses stark die Flexibilität des Audit Dienstes.

6.6.4 Non-Repudiation (Unabstreitbarkeit)

Diese Regelung verhindert, dass ein Teilnehmer (principal) seine Aktionen in einem verteilten System erfolgreich abstreiten kann, wenn dieser diese Aktionen tatsächlich ausführte.
Die Spezifikation des non-repudiation Dienstes innerhalb der CORBAsec ist ebenfalls unvollständig. Es werden die Komponenten delivery authority und secure storage benötigt. Die Spezifikation referenziert zu anderen Diensten und Standards, erläutert aber nicht explizit die Details des non-repudiation Dienstes zum Thema Integration und Standard. Zu diesem Zeitpunkt existieren keine CORBA Produkte, die den ISO Non-Repudiation Framework beinhalten. Dieses ist aber die Basis für den CORBAsec non-repudiation Dienst. Zudem ist keine Prototyp Implementierung für den Standard verfügbar.
Daneben bieten die unterstützten Schnittstellen nur eine Untermenge von verschiedenen Arten von evidence an. Es ist nur möglich zu beweisen, dass eine Nachricht gesendet oder empfangen wurde, aber nicht, ob diese erfolgreich ausgeführt wurde. Nur ein vertrauensvoller non-repudiation Dienst auf der ORB Schicht kann diese Funktionalität erreichen.
Da kein Standard Format für non-repudiation Tokens existiert, ist es unmöglich, eine Interoperabilität sicherzustellen. Der Hauptgrund liegt daran, dass keine Vorschläge für eine einheitliche Technologie existieren.
Der non-repudiation Dienst wurde als stand-alone ersetzbarer Dienst angekündigt, allerdings zeigt dieser Dienst starke Abhängigkeiten zu anderen Komponenten auf, wie z.B. dem CORBA Time-Service. Sowohl asymmetrische und symmetrische Kryptographie können genutzt werden, diese Methoden sind zwar in sich selbst ersetzbar (austauschbar), allerdings nicht der Dienst selbst.
Es existieren noch weitere Gründe für das Fehlen von Implementierungen des CORBA non-repudiation Dienstes. Zunächst basiert der CORBAsec non-repudiation Dienst auf dem Sicherheitslevel 2 und es sind nur wenige Sicherheitslevel 2 Produkte auf dem Markt. Zweitens müssen die CORBA Produkthersteller nicht den non-repudiation Dienst implementieren, da dieser Dienst nur als optional dekla-

riert wurde. Außerdem verhindern verschiedene juristische Einschränkungen die Weiterentwicklung des non-repudiation Dienstes.

6.6.5 Assurance

Mit Assurance wird die Zuversicht in die Sicherheit eines Systems beschrieben.
Im Gegensatz zu stand-alone Systemen ist es sehr kompliziert, Vertrauen (trust) in verteilten Systemen herzustellen, da viele verschiedene Komponenten und Mechanismen in diesem Prozess involviert sind. Zudem ändern sich Vertrauensverbindungen häufig, welches das Prinzip des assurance schwer macht. Die CORBAsec führte das Prinzip des Distributed Trusted Computing Base ein, welches alle sicherheitskritischen Komponenten beinhaltet. Dazu gehören Anwendungsobjekte, die die Sicherheit durchsetzen, der ORB Kernel, object adaptersm security interceptors. In der Praxis ist nur oft eine teilweise Analyse aller Komponenten möglich, da viele Komponenten nicht unter der Verantwortung eines einzigen Herstellers stehen.

6.6.6 Interoperabilität

Eigentlich sollte CORBA Hauptstärke dessen Plattform und Programmiersprachen-Unabhängigkeit sein, wo verteilte Anwendungen auf mehreren unterschiedlichen Maschinen laufen können, sogar auf unterschiedlichen ORB Implementierungen. Es ist aber dem Produkthersteller überlassen, die das Niveau der Interoperabilität zu steuern.

6.7 Schlussfolgerung

Die CORBA Security Service definiert in seiner Spezifikation eine Vielzahl Mechanismen zur Durchsetzung von Policies. Softwaretechnisch spezifiziert CORBASec nur Interfaces, die entweder vom ORB oder der Anwendung selbst implementiert werden müssen. Aus diesem Grund existieren noch wenige CSI 2 Level ORB Realisierungen, außerdem weisen die unterschiedlichen ORBs im Zusammenhang mit Sicherheit Inkompatibilitäten auf. Zudem ist nicht immer klar, welche Schicht eine Policy durchsetzen soll.
Das CORBA Sicherheit Modell ist ein sicherheitstechnikneutrales Framework. Schnittstellen, die für die Sicherheit von client-target object invocation verwendet werden, kapseln die Sicherheitsmechanismen, die von den Anwendungsobjekten und dem ORB genutzt werden.
Es ist möglich, CORBA Sicherheit auf einer breiten Vielzahl der vorhandenen Systemen einzuführen. Damit werden die Sicherheitsprotokolle wieder verwendet, die schon nativ zu einem System gehören.
Der CORBA Sicherheit Service kann Zugang zu Anwendungs- Objekten realisieren, ohne dass das die Anwendung dieses merkt. Damit kann der Service auf andere Systeme portiert werden, die andere Security Policies durchsetzen möchten.
Diese Spezifikation definiert den Kernsicherheit Service und die Schnittstellen, die gefordert werden, um ein angemessenes Niveau an Sicherheit eines CORBA-konformen System als Ganzes sicherzustellen.

Literaturverzeichnis

[1] Beznosov. Overview of CORBA Security, 2001.

[2] Brose. Access Control in CORBA. TU-Berlin, 2001.

[3] Lacoste. Towards a Secure Platform for Distributed Mobile Object Computing. France Telecom, 2001.

[4] OMG. Security Service Specification 1.8, 2002.

7 Trust Management und Security Policy Enforcement am Beispiel zweier großer Datenbankmanagementsysteme - Oracle und MS SQL-Server

Kolja Engelmann, Lisa Böhringer

Abstract

Eine Datenbank ist eine zusammenhängende, integrierte Datensammlung untereinander verknüpfter Daten für den Multi-User Betrieb und eigenem Verwaltungssystem, dem Datenbankmanagementsystem (DBMS).[1] Eine der wichtigsten Aufgaben eines DBMS, neben der effizienten Verwaltung und dem schnellen Zugriff auf Daten, ist der Datenschutz, was den Schutz der gespeicherten Informationen gegen einen unbefugten Zugriff durch Dritte bedeutet. Hierbei gilt es sowohl die gespeicherten Informationen, als auch die Datenbank an sich durch Trust Management und Policy Enforcement zu schützen.
Die Möglichkeiten mittels derer Datenbankmanagementsysteme Datensicherheit erreichen wollen, sind sehr vielfältig und sollen in der vorliegenden Ausarbeitung anhand zwei der größten momentan kommerziell verfügbaren Datenbankmanagementsysteme, Oracle DBMS 9i und MSSQL Server 2000, auszugsweise dargestellt werden.[2]

7.1 Einleitung

„*A database security policy is an implementation of an overall system security policy, which in turn is often derived from a broad, organizational security policy.[. . .]*“[3]
Datenbanksicherheit ist ein sehr breites Gebiet, das viele Problemkreise abdecken muss. Hierzu zählen unter anderem:

- Juristische und ethische Faktoren in Bezug auf das Recht für den Zugang zu bestimmten Informationen

[1] Christian Bänsch, Relationale Datenbanken, Grundkurs Informatik, Bertha – von - Suttner - Gymnasium, 2000
[2] Zwar existiert bereits das Oracle DBMS in der Version 11i, da jedoch eine aktuelle Dokumentation hierfür nicht zugänglich war, müssen wir uns auf die Vorstellung der Version 9i beschränken
[3] Oracle Label Based Security, Administrators Guide Release 2(9.2), März 2002, Seite 33, Absatz 3

- Technische Faktoren, die auf der Hardware- und Betriebssystem- oder DBMS-Ebene umgesetzt werden sollen
- Die Notwendigkeit mehrere Sicherheitsebenen(z.B. streng geheim, geheim, vertraulich, unklassifiziert) zu erstellen und auf Basis dieser Ebenen Benutzern zu kategorisieren

Ein DBMS muss folglich Techniken bereitstellen, um definierten Benutzern oder Benutzergruppen den Zugriff auf ausgewählte Daten und Funktionen einer Datenbank zu gewähren und dabei den Zugriff auf den Rest der Datenbank zu verwehren. Die zentrale Autorität für die Verwaltung diese Aufgabenbereiche ist der Datenbankadministrator, der DBA. Dieser Superuser ist verantwortlich für die Erstellung von Useraccounts, und Usergruppen, also der Zugriffskontrolle auf das DBMS als Ganzes, der Vergabe von Privilegien an Benutzer und Gruppen, was ,,Autorisierung nach eigenem Ermessen“ (Discretionary Access Control - DAC) genannt wird und für die Einstufung von Benutzern und Daten in übereinstimmung mit gegebenen Sicherheitsvorschriften, was ,,Verbindlich vorgeschriebene Sicherheitsmechanismen“(Mandatory Access Conrol - MAC) genannt wird. Ferner ist es die Aufgabe des DBA, mittels der DBMS eigenen Möglichkeiten, eine Reihe von Sicherheitsbedrohungen zu minimieren.
Im Rahmen dieser Ausarbeitung möchten wir zunächst auf die gebräuchlichsten Sicherheitsbedrohungen eingehen, um für die möglichen Probleme zu sensibilisieren. Es werden die Lösungsansätze DAC und MAC, die bereits in der SQL'92 Ableitung SQL2 vorgestellt wurden, behandelt, da diese die theoretische Grundlage für die Implementationen der anschließend betrachteten DBMS Oracle 9i und SQL Server 2000 sind. Abschließend möchten wir einen Blick auf die neuesten Versionen der betrachteten DBMS werfen, nämlich Oracle 11i und SQL Server 2005 – Yukon.

7.1.1 Security Threats

Die zunehmende Menge an zu speichernden Daten in einer verteilten Umgebung und die Komponente Mensch als Benutzer, konfrontiert Datenbankadministratoren mit einer Reihe von vielfältigen Sicherheitsbedrohungen. Auszugsweise seien hierbei die folgenden Punkte genannt und anschließend kurz erklärt.

- Datenverfälschung
- Datendiebstahl und Abhören von Daten auf dem übertragungsmedium
- Arbeiten unter falscher Benutzeridentität
- Passwort-Bedrohungen jeglicher Art

Was die verglichenen DBMS gegen diese Sicherheitsbedrohungen schlussendlich zu unternehmen versuchen, wird an geeigneter Stelle von uns vorgestellt.

Datenverfälschung

Der im Umgang mit großen Datenbanksystemen häufig gegebene Umstand der verteilten Position von Datenbank und Anwendungslogik ermöglicht potentiellen Angreifern die Datenintegrität während der übertragung auf dem übertragungsmedium

zu verletzen und somit zu inkonsistenten Datenbankzuständen zu führen. Dies gefährdet einen der Grundpfeiler des Datenbankenprinzips[4].

Datendiebstahl und Abhören von Daten auf dem Übertragungsmedium

Sowohl im Intra-, als auch im Internet gibt es keine sicheren übertragungswege. Ein potentieller Angreifer mit Zugang zum übertragungsmedium ist stets in der Lage, mit mehr oder weniger Aufwand übertragene Daten abzuhören.

Arbeiten unter falscher Benutzeridentität

In einer verteilten Multiuser Umgebung ist es schwer bis unmöglich festzustellen, ob ein Benutzer der ist, für den er sich ausgibt oder ob eine Anfrage wirklich von einem angegebenen Ausgangspunkt stammt.

Passwortbedrohungen

Benutzer verwenden häufig einfach zu merkende Passwörter, welche schnell Ziel einer Wörterbuchattacke werden können.

7.2 Allgemeine Techniken zur Erzeugung von Datensicherheit

7.2.1 Zugriffsschutz durch Benutzer-Accounts

Möchte ein Benutzer oder eine Gruppe auf ein Datenbanksystem zugreifen, so muss dafür ein gültiger Benutzer-Account vorhanden sein. Dieser wird meist in einer verschlüsselten Tabelle des DBMS gespeichert, oder durch das umgebende Betriebssystem bereitgestellt, von wo aus sie einfach zu warten und auszuwerten sind. DBMS spezifische Arten der Verifizierung dieser Accounts werden an geeigneter Stelle vorgestellt.

7.2.2 Discretionary Access Controll (DAC)

„*Die typische Methode zur Umsetzung einer DAC in einem Datenbanksystem basiert auf der Vergabe und dem Widerruf von Privilegien.[...]Das wichtigste Konzept ist dabei die Einbeziehung zusätzlicher Anweisungen in die Anfragesprache, die es dem DBA und ausgewählten Benutzern ermöglichen, Privilegien zu gewähren und zu widerrufen.*“[5]
Das DBMS muss dazu einen selektiven Zugriff für jede Relation und Operation auf Grundlage spezifischer Accounts bereitstellen. Informell wird in Bezug auf die Zuweisung von Privilegien zur Nutzung des Datenbanksystems zwischen der Account-Ebene und der Relations-, oder Tabellenebene unterschieden.

[4] ACID Prinzip C-Consitency, eine Datenbank muss immer in einem konsistenten Zustand sein.
[5] Ramez Elmasri,Shamkant B. Navathe, Grundlagen von Datenbanksystemen, Addison-Wesley, 3.überarbeitete Auflage, 2002 – Seite 771 Absatz 1

Account Ebene

Die Privilegien auf der Account-Ebene werden unabhängig von den Relationen der Datenbank vergeben und betreffen die Berechtigung selbst neue Schemata oder Basisrelationen zu erstellen (CREATE TABLE, CREATE SCHEMA, CREATE VIEW), diese zu verändern (ALTER) oder zu löschen(DROP).

Relations-, oder Tabellenebene

Auf der Relations-, oder Tabellenebene werden Privilegien für Basisrelationen oder virtuelle Relationen(Views) spezifiziert. Privilegien auf Relationsebene spezifizieren für jeden Rechteinhaber die Relationen, auf die Befehle ausgeführt werden dürfen. Die Vergabe und der Widerruf dieser Privilegien basiert allgemein auf einem DAC Autorisierungsmodell, das als Zugriffsmatrixmodell bezeichnet wird, wobei die Zeilen der Matrix Subjekte (Benutzer, Accounts, Programme) und die Spalten Objekte (Relationen, Datensätze, Views, Operationen) darstellen. Jede Position $M_{i,j}$ stellt die Menge von Privilegien dar, die Subjekt i auf Objekt j besitzt.

Tabelle 7.1: Zugriffsmatrixmodell

Subjekt	**Relation A**	**View A**	**Operation A**
Account 1	Select,Update,Delete	Select	execute
Account 2	Select	Select, Update	-

Propagieren von Zugriffsrechten

Der Initiator jeglicher Rechtevergabe ist der DBA. Ihm obliegt es Privilegien an Nutzer und Gruppen zu verteilen (*GRANT*) und auch diesen zu ermöglichen (*GRANT[...] with GRANT OPTION*), die erhaltenen Rechte an andere User oder Gruppen weiterzugeben, wieder zu entfernen(REVOKE) oder zu verbieten (DENY). Die Rechtevergabe ist dabei transitiv. Erteilt A ein Privileg an B und erlaubt B außerdem dessen Propagierung, so kann B dieses Privileg seinerseits an C geben. Wird nun A selbst das Privileg wieder entzogen, so verfällt es auch bei allen Subjekten, die es zuvor auf direktem oder indirektem Wege von A erhalten haben. (s. Tabelle 7.1)

7.2.3 Mandatory Access Control

Bei Discretionary Access Control handelte es sich um den traditionell wichtigsten Sicherheitsmechanismus für relationale Datenbanksysteme, eine ,,Alles-oder-Nichts-Methode“[6]. Die Mandatory Access Control ist eine zusätzliche Sicherheitsvergabe und erweitert die DAC um Sicherheitsklassen. Alle Benutzer, Daten und Relationen sind Element einer Sicherheitsklasse, für die individuelle Rechte gelten.

[6] Ramez Elmasri,Shamkant B. Navathe, Grundlagen von Datenbanksystemen, Addison-Wesley, 3.überarbeitete Auflage, 2002 – Seite 776 Absatz 1 - Entweder der Zugriff ist erlaubt, oder aber nicht

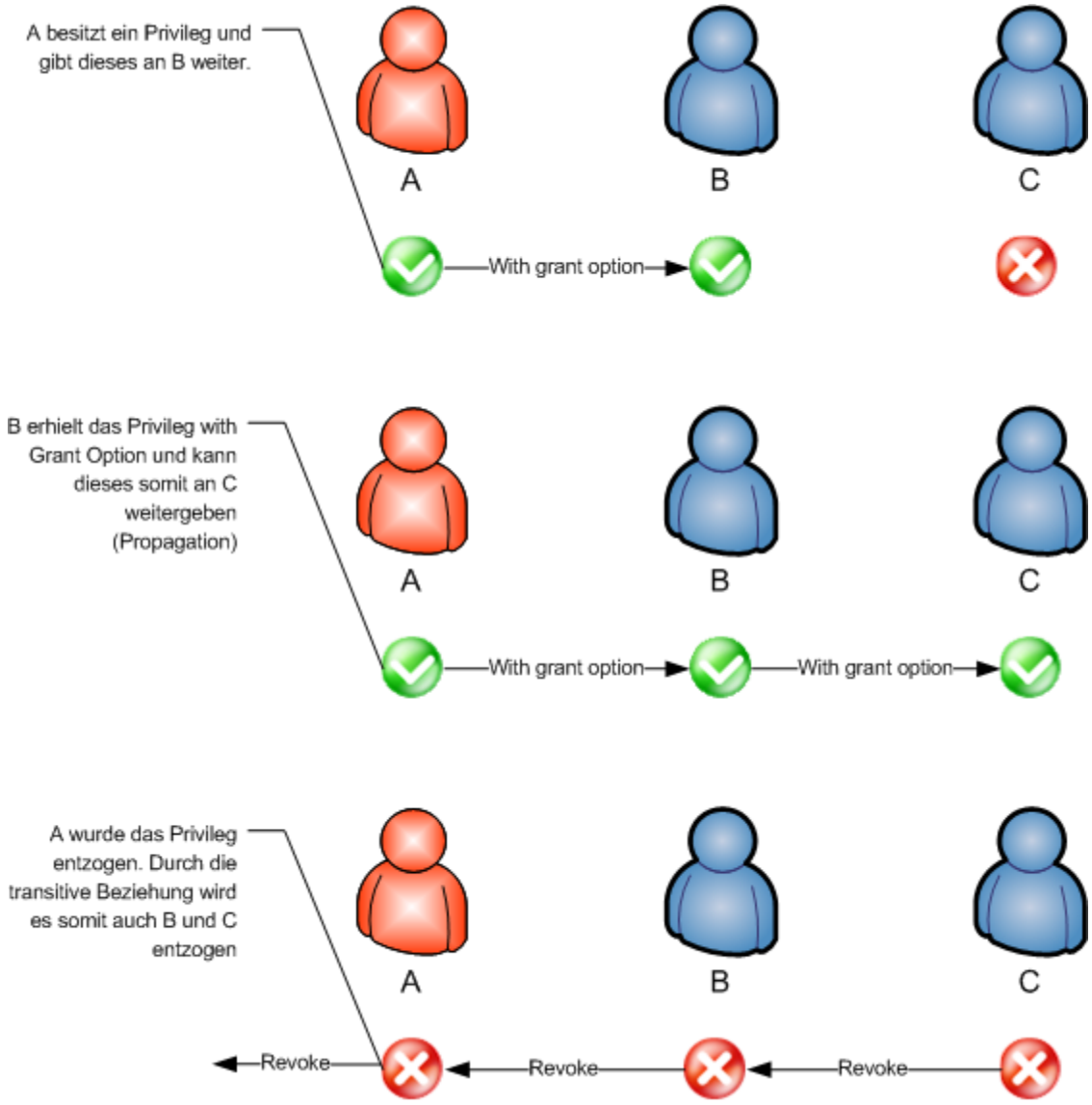


Abbildung 7.1: Schematische Darstellung der transitiven Rechtevergabe

Das hierfür zu Grunde liegende Modell ist das Bell-LaPadula-Modell, welches die Sicherheitsklassen als Teilmengenbeziehung sieht.

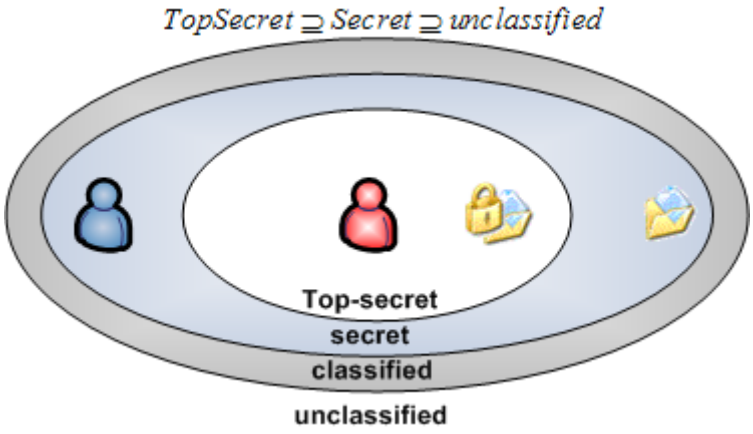


Abbildung 7.2: Bell-Lapadula Modell

Auf Grundlage der Subjekt/Objekt-Einstufungen (s. Seite4) werden in Bezug auf den Datenzugriff zwei Einschränkungen auferlegt.

1. Ein Subjekt S erhält nur Lesezugriff auf ein Objekt O, wenn es in dessen Sicherheitsklasse liegt oder Element einer Teilmenge der Sicherheitsklasse von O ist (einfache Sicherheitseigenschaft)

2. Ein Subjekt S darf ein Objekt O nicht schreiben, wenn O auf einer niedrigeren Sicherheitsstufe als die Ermächtigungsstufe von S liegt (*-Eigenschaft), was verhindern soll, dass Objekte höherer Sicherheitsstufe in Bereiche niedrigerer Sicherheitsstufe übertragen werden können

Um dieses Sicherheitskonzept in relationale Datenbanksysteme zu integrieren, muss jedem Attributwert eines Tupels und jedem Tupel als Ganzes ein Einstufungsattribut zugeordnet (s. Abbildung 7.1). Das Einstufungsattribut des gesamten Tupels stellt hierbei die allgemeine Einstufung dar, jedes Attribut muss ein kleineres oder gleich starkes Einstufungsattribut besitzen. Dieser effiziente und stark skalierbare Sicherheitsmechanismus existiert zwar bereits im SQL 2 Standard von 1992, wurde aber bisher in noch keinem DBMS integriert[7].

7.3 Oracle

7.3.1 Oracle Security Mechanismen

Die von Oracle „Oracle Advanced Security“ getaufte Menge an Sicherheitsvorkehrungen bietet verschiedene Ansätze, um das Ziel der Datensicherheit weitestgehend zu erreichen. Oracle führt hierzu die folgenden Punkte auf:

- Datenschutz

[7] Es wurden nur ähnliche Verfahren entwickelt, so zum Beispiel Virtual Privat Database und Label Based Security bei Oracle

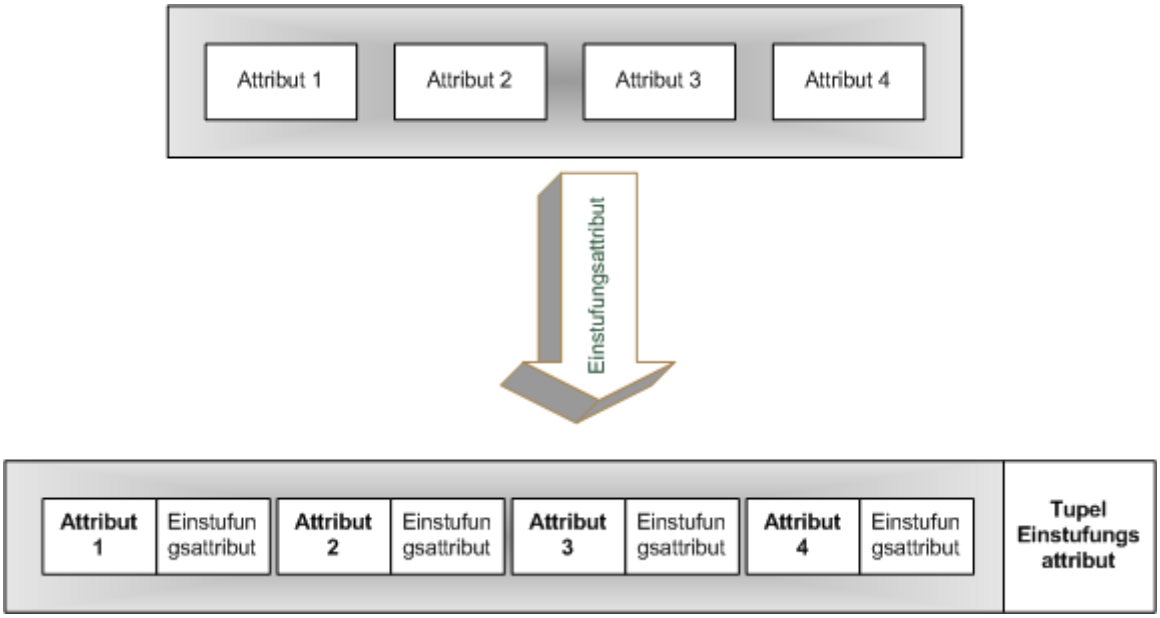


Abbildung 7.3: Einstufungsattribute für Sicherheitsklassen

- Datenintegrität
- Authentifizierung
- Autorisation

„Oracle Advanced Security“ ist ein Add-On, welches die Oracle Client oder Server Installation ergänzt. Die verwendeten Module werden dabei für die aufrufenden Applikationen transparent genutzt, es ist keine änderung an der Applikation selbst notwenig, wenn beispielsweise der verwendete Verschlüsselungsalgorithmus ausgetauscht wird(s. Abbildung 7.2).

Datenschutz

Unter Datenschutz versteht Oracle die Verschlüsselung von Datenübertragungen mittels starker Verschlüsselungsalgorithmen. Dank der Lockerung der amerikanischen Exportrichtlinien für diese Algorithmen, ist es Oracle möglich, sein DBMS mit mehreren Algorithmen auszustatten,[8] welche vom DBMS Administrator frei gewählt werden können und zur Folge haben, dass das Oracle DBMS seit der Version 8i in der „Federal Information Processing Standard“-Klasse 2 eingestuft ist.
„*Oracle Advanced Security Release 8.1.6 has been validated under U.S. Federal Information Processing Standard 140-1 (FIPS) at the Level 2 security level. This provides independent confirmation that Oracle Advanced Security conforms to federal government standards.*”[9] Folgende Verschlüsselungsalgorithmen werden von Oracle durch austauschbare Verschlüsselungsmodule angeboten.

[8] Reinhard Wobst, Abenteuer Kryptologie – Methoden, Risiken und Nutzen der Datenverschlüsselung, Addison-Wesley, 2001, 3. überarbeitete Auflage
[9] Oracle. Advanced Security, Administrator's Guide,Release 2 (9.2), March 2002 Seite 45

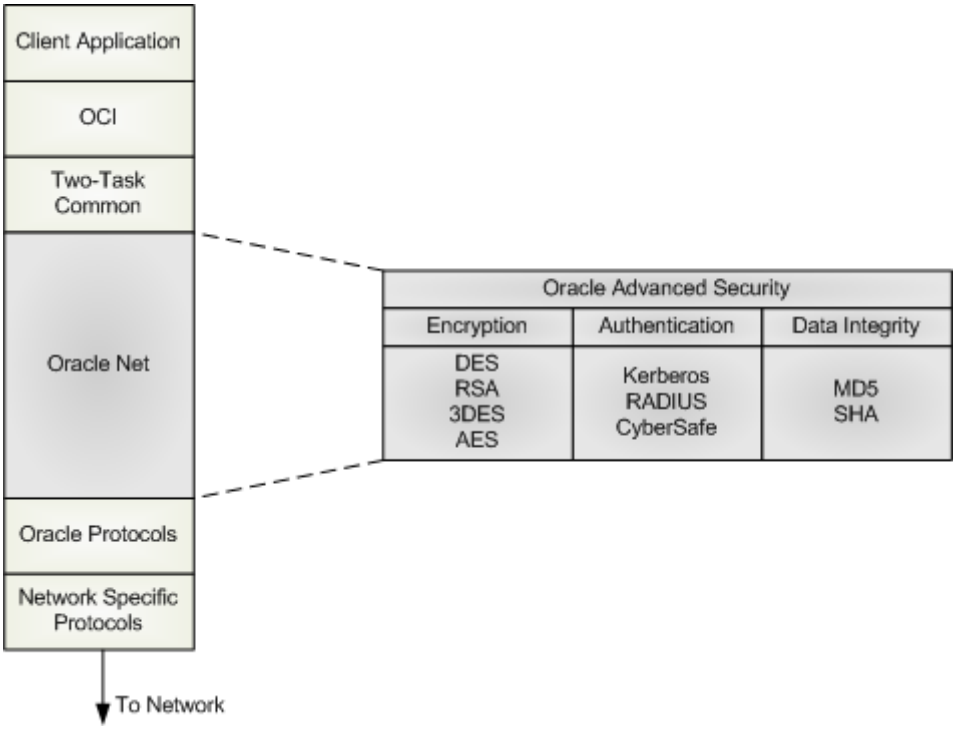


Abbildung 7.4: Oracle Advanced Security in Oracle Networking Environment

RC4 Das RC4 Verschlüsselungsmodul verwendet den RSA Security, Inc. RC4 Algorithmus. Mittels des generierten Sitzungsschlüssels ist es möglich jeden Netzwerkverkehr zu sichern, wozu alle Daten, SQL Statements, sowie Stored Procedures Aufrufe und Ergebnisse gehören. Der hohe Sicherheitsgrad mit unterstützten Schlüssellängen von 40bits, 56bits, 128bits und 256bits geht jedoch auf Kosten der Geschwindigkeit.

DES Verschlüsselung Der amerikanische Data Encryption Standard benutzt ein symmetrisches Verschlüsselungsverfahren, und wird in einer speziell angepassten Version mit 56bit Schlüssellänge von Oracle verwendet, um die Datenkommunikation zu verschlüsseln. Oracle liefert außerdem ein DES40 Modul, welches nur 40bit Schlüsselstärke verwendet, um kompatibel zu bleiben.

Triple-DES Verschlüsselung 3DES verschlüsselt Nachrichten mit drei DES Verschlüsselungsdurchgängen.
Oracle implementiert 3DES mit effektiven Schlüssellängen von 112bits und 168bits in einer zwei-Schlüssel und einer drei-Schlüssel Variante, die beide im Cipher-Block Chaining Mode[10] arbeiten. Der Nachteil liegt in der dreimal längeren Verschlüsselungszeit gegenüber DES.

AES ,,*Das NIST entwickelt in Zusammenarbeit mit Industrie-Unternehmen seit Jahren den AES-Verschlüsselungsstandard. Diese symmetrische Verschlüsselung*

[10] Vor der Chiffrierung eines Klartextblocks wird dieser mit dem im letzten Schritt erzeugten Geheimtextblock per XOR verknüpft.

soll den bisherigen DES-Standard ablösen. Der AES-Standard spezifiziert drei verschiedene Schlüsselgrößen mit 128, 192 und 256 Bit.[...]“[11]. AES128, AES192 und AES256 arbeiten im outer-CBC mode.

Datenintegrität

Um die Integrität der Daten während einer übertragung zu sichern, werden von Oracle Hashwerte über die zu sendenden Nutzdaten mittels MD5 oder SHA-1 generiert und mit jeder versendeten Nachricht übertragen. Der zusätzlich generierte Overhead bei jeder übermittelten Nachricht wird jedoch akzeptiert, da hierdurch ein Schutz gegen Datenmanipulation, gelöschte Pakete und Replay Attacken erkauft wird.

Authentifizierung

Die Userauthentifizierung in einem verteilten Datenbanksystem ist unabkömmlich für die Sicherheit und wird meist mittels einer Passwortabfrage realisiert. Den generellen Ablauf einer zentralisierten Authentifizierung beschreibt Oracle, wie es in der folgenden Grafik schematisch dargestellt wird.
Oracle bietet in seinem DBMS verschiedene Möglichkeiten der Authentifizierung an. Einerseits kann die Authentifizierung über SSL und digitale Zertifikate erfolgen, oder aber über Services anderer Anbieter. Als Module sind in Oracle 9i die folgenden Authentifizierungsmethoden integriert.

- Secure Socket layer mit digitalen Zertifikaten
- Entrust/PKI
- Remote Authentication Dial-In User Service (RADIUS)
- Kerberos / Cyber Safe
- Smart Cards

Diese dienen Oracle dazu eine ,,Single Sign-On“ genannte Authentifizierungsstrategie durchzuführen. ,,Single Sign-On“ ermöglicht es einem Benutzer sich einmalig am System anzumelden und alle weiteren Authentifizierungswünsche automatisiert über die bereits erfolgte Anmeldung ablaufen zu lassen.

Secure Socket Layer Oracle setzt SSL dazu ein entweder einen Client nur gegenüber einem Server zu authentifizieren, einen Server gegenüber einem Client zu authentifizieren oder beides. SSL wird von Oracle entweder allein, oder in Kombination mit anderen Authentifizierungsverfahren verwendet, je nach Einstellung durch den DBMS Administrator.

Entrust/PKI Oracle unterstützt die Public Key Infrastructure der Firma Entrust Technologies, Inc. und ermöglicht es das Single Sign-On von Entrust zur Authentifizierung zu verwenden.

[11] Entnommen aus dem Siemens Online Lexikon http://www.networks.siemens.de/solutionprovider/_online_lexikon/0/f011410.

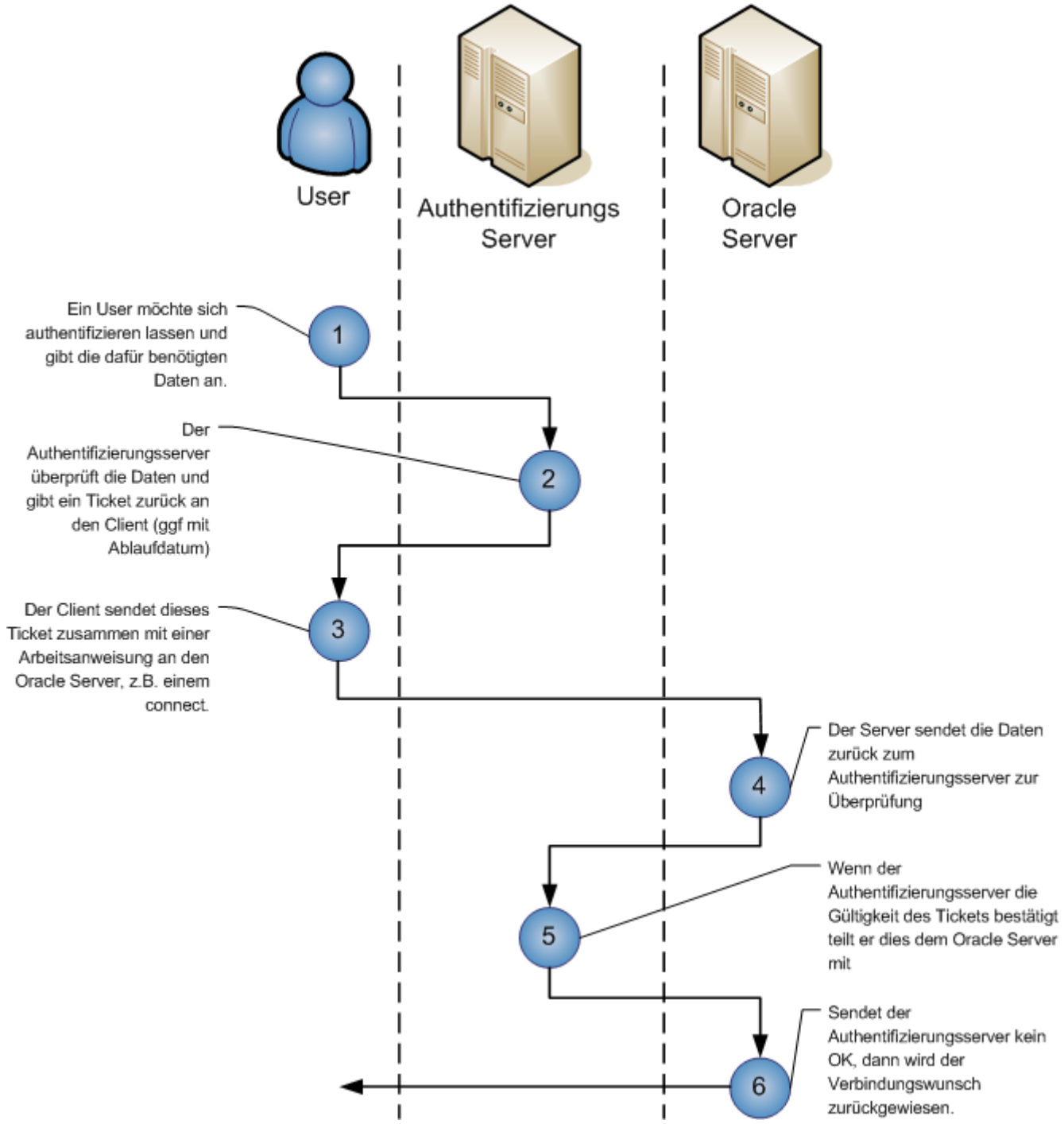


Abbildung 7.5: Authentifizierungsablauf Oracle

Remote Authentication Dial-In User Service (RADIUS) RADIUS ist ein standardisiertes Client-Server-Protokoll zur entfernten Authentifizierung von Usern. Oracle unterstützt RADIUS, da es von einer Reihe von Authentifizierungsmechanismen ebenfalls verwendet wird, z.B. von Smart Cards.

Kerberos/Cybersafe [12]Die von Oracle implementierte Variante der Kerberos Authentifizierung unterstützt eine Datenbank – Verbindung – Authentifizierung, was nicht einmal von Cybersafe, einer kommerziellen Implementierung von Kerberos unterstützt wird.

Smart Cards Eine RADIUS konforme Smart Card ist eine mit Speicher und Prozessor bestückte Karte, die von einem Kartenlesegerät des Clients gelesen wird, um eine Authentifizierung durchzuführen.

Autorisation

Die Autorisierung von Benutzern in Oracle wird durch die Zuweisung von Rollen und Privilegien realisiert, allerdings stark von den Authentifizierungsmethoden, die von Oracle bereitgestellt werden, erweitert. So ist es Beispielsweise unter Unix möglich, die Benutzerautorisation mittels Distributed Computing Environment (DCE) durchzuführen oder Benutzer und Rechte zentral mittels LDAP zu verwalten.

7.3.2 Label Based Security

„*In Oracle Label Security you can apply different enforcement options for maximum flexibility in controlling the different Data Manipulation Language operations that users can perform. For each operation—SELECT, INSERT, UPDATE, and DELETE—you can specify a particular type of enforcement of the security policy, for each table. In this way you can customize the label-based access controls on each table..*“[13]
„Label Based Security“ ist ein von Oracle entwickeltes Verfahren zur Implementierung von Mandatory Access Control (s. Mandatory Access Control), auf Basis von Virtual Private Databases (VPD). Es ermöglicht die Zugriffskontrolle auf gespeicherte Daten über ein jedem Tupel hinzugefügtem Label, wodurch sich das Tupel in eine oder mehrere Sicherheitsgruppen einordnen und Zugriffsbeschränkungen zuordnen lässt. Für jede abgesetzte Anfrage auf die Datenbank werden die Labels nun ausgewertet und als Ergebnis erscheint eine VPD. Das Beispiel (s. Abbildung 7.3) zeigt dies anhand zweier Tabelscans auf eine Tabelle „Orders“ durch User in unterschiedlichen Sicherheitsgruppen.
[14]

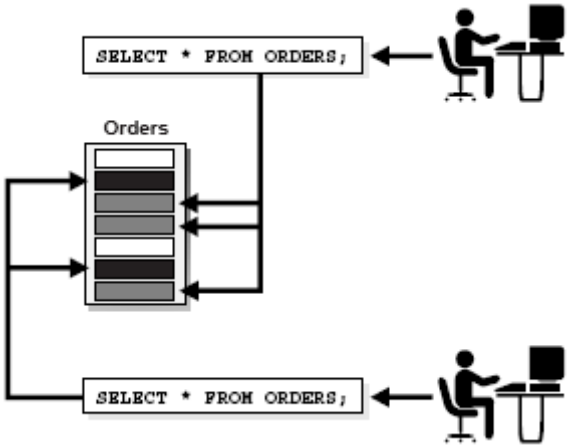


Abbildung 7.6: Virtual Private Database

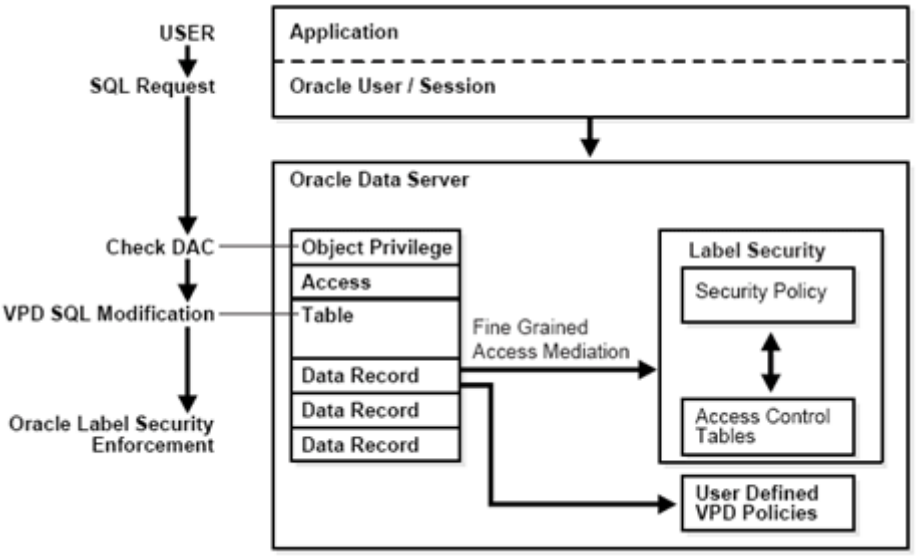


Abbildung 7.7: Ablauf einer Anfrage gegen eine VPD und LBS geschützte Tabelle

Ablauf einer ,,Label Based Security“ Anfrage

[15]

Bevor eine Anfrage ausgeführt wird, werden die DAC Privilegien des ausführenden Benutzerkontos überprüft, ob dieses auch die nötigen Rechte aufweist. Bei Erfolg wird überprüft ob eine VPD Policy für die betreffenden Tabellen vorhanden ist und die Query ,,on the fly“ verändert, so dass nur Tupel zurückgegeben werden, die der Sicherheitspolicy entsprechen.

Speicherung von Labels in der Tabellenstruktur

Um ,,Label Based Security“ effizient speichern und Anfragen auf Basis der Labels schnell berechnen zu können, speichert Oracle diese intern als Bitmap Index ab. Diese sind eine Form des Index, die besonders effizient ausgewertet werden kann. Für jeden möglichen Wert, den der Index annehmen kann, wird eine Spalte in der Indextabelle definiert und ein binärer Wert eingetragen. 1 – Das Label des Tupels besitzt diesen Wert, oder 0, das Label des Tupels besitzt diesen Wert nicht (s. Abbildung 7.4). Effizient ist diese Darstellung gerade deshalb, weil der Index sehr wenig Speicher verbraucht und deswegen komplett im Hauptspeicher liegen kann. Alle Operationen sind binärer Natur und somit im Prozessor in Hardware implementiert.

Abbildung 7.8: Bitmap Index

[12] Entnommen aus dem Siemens Online Lexikon http://www.networks.siemens.de/solutionprovider/_online_lexikon/9/f008959.
[13] Oracle Label Based Security, Administrators Guide Release 2(9.2), März 2002, Seite 41, Absatz 3
[14] Oracle Label Based Security, Administrators Guide Release 2(9.2), März 2002, Seite 37
[15] Oracle Label Based Security, Administrators Guide Release 2(9.2), März 2002, Seite 38

7.4 MS SQL Server 2000

7.4.1 MS-SQL Allgemein

Der SQL Server 2000 ist ein relationales Client-Server-Datenbank-Managementsystem (RDBMS).
Das Haupteinsatzgebiet des SQL Server 2000 liegt vor allem in Datenbankanwendungen, Datawarehousing und Webanwendungen, bei denen mehrere verschiedene Benutzer parallel auf große Datenmengen zugreifen, um diese zu ver- oder bearbeiten.

7.4.2 C2 und Sicherheitsüberprüfung

C2-Zertifikat

Der SQL Server 2000 erfüllt das C2 Sicherheitszertifikat, welches von der US Regierung spezifiziert wurde und als eines der besten Sicherheitszertifikate gilt. Schon in den 80er Jahren hat das amerikanische Verteidigungsministerium die "Trusted Computer Systems Evaluation Criteria"(TCSEC) veröffentlicht, die wegen ihres Einbandes als Örange Book"bekannt wurden. In diesen TCSEC wurden sieben Sicherheitsstufen festgelegt: D, C1, C2, B1, B2, B3 und A1, wobei D die niedrigste und A1 die höchste Sicherheitsstufe darstellt. Diese Stufen wurden vom Bundesamt für Sicherheit in der Informationstechnik in leicht abgeänderter Form übernommen. Je höher die Sicherheitsstufe, desto umfangreicher die Sicherheitsmerkmale. Damit ein Produkt C2 erfüllen kann, müssen u.A. folgende Kriterien erfüllt sein:

1. Jeder Benutzer muss sich vor der Benutzung eines Rechners mit seiner Benutzerkennung und seinem Passwort identifizieren und authentisieren

2. Es muss eine detaillierte Rechteverwaltung auf Datei- und Verzeichnisebene für einzelne Benutzer bzw. Benutzergruppen möglich sein. Die Rechteverwaltung muss derart gestaltet sein, dass der Zugriff gänzlich verweigert werden kann

3. Die Rechte- und die Benutzerverwaltung dürfen nur durch autorisierte Benutzer erfolgen

4. Bei jedem Zugriff muss die Berechtigung überprüft werden. Unberechtigte Zugriffe müssen abgewiesen werden

5. Alle Aktionen innerhalb einer Sitzung, auch die von autorisierten Benutzern, müssen protokolliert werden können (Auditing). Die Protokolle dürfen nur autorisierten Benutzern zugänglich sein. Die Programme zur Auswertung der Protokolle müssen vorhanden und dokumentiert sein. Es muss möglich sein, einzelne Aktionen einzelner Benutzer oder Gruppen zu filtern

6. Vor der Benutzung durch einen anderen Benutzer muss der Speicherinhalt so aufbereitet werden, dass keine Rückschlüsse auf den früheren Inhalt möglich sind

Das C2-Zertifikat des SQL Servers gilt nur für eine stand-alone-Workstation. Sobald eine Workstation in ein Netz eingebunden wird, erlischt das Zertifikat. Das Zertifikat ist auch nur dann gültig, wenn das Dateisystem NTFS benutzt wird.
Um der Auditing Vorgabe des C2 Zertifikates gerecht zu werden, ist in den SQL Server 2000 standardmäßig ein voll funktionstüchtiger Prüfmechanismus eingebaut, welcher aus mehreren Komponenten besteht. Hierzu zählen u.A. die SQL Trace Komponente und die SQL Profiler Komponente.

SQL Trace

Unter SQL Trace versteht man die Möglichkeit mit Hilfe eines Programms (Traceprogramms) Verbindungen mit dem SQL Server 2000 aufzeichnen zu können. Wird der Umfang der Beobachtungen zu groß, so kann dieser über Filter eingeschränkt werden. Es erfolgt also eine Unterstützung bei der überwachung der Performanceinformationen des SQL Servers 2000.

SQL Profiler

Der SQL Profiler ist ein Client des SQL Trace. Es handelt sich hierbei um ein grafisches Hilfsmittel, welches Daten sichtbar, überprüfbar und nachvollziehbar macht und zusätzlich das Ausführen von ausgewählten Aktionen auf den Daten ermöglicht.

7.4.3 Kerberos und Delegation

In der Windows 2000 Umgebung wird Kerberos als primärer Authentifizierungsmechanismus verwendet. Durch dessen Single Sign-On Konzept, lässt sich die Authentifizierung eines Users im Windows System gleichzeitig als Anmeldung an das Microsoft DBMS verwenden, wie es auch bei Oracle genutzt wird. Der SQL Server 2000 unterstützt dabei nicht nur die Userauthentifizierung von Kerberos, sondern auch dessen Eigenschaften delegierte Aufgaben zu akzeptieren und weiterzuleiten. Um allerdings Aufgaben an andere Rechner delegieren zu können, muss auf allen beteiligten Rechnern Windows 2000 (oder höher) und Kerberos installiert sein.

7.4.4 Datenschutz

SSL und TLS

Der SQL Server 2000 verschlüsselt Daten anhand von auf dem Server installierten Verschlüsselungszertifikaten. Die Stufe der Verschlüsselung hängt von den vorhandenen Zertifikatsbestimmungen, ebenso wie von den kryptografischen Fähigkeiten des Clients und des Servers ab. Ist ein Zertifikat vorhanden und ausführbar, so werden während eines Logins alle verwandten Datenpakete automatisch verschlüsselt. Anschließend können alle Request von und an den Server ebenfalls verschlüsselt werden.

Verschlüsselung des Client-Request Der Client hat die Möglichkeit den gesamten Datenverkehr zwischen ihm und den Server zu verschlüsseln. Des Weiteren verhindert die Client-Request-Verschlüsselung den Zugang zu SQL Server 2000 Servern, welche kein gültiges Zertifikat besitzen.

Verschlüsselung des Server-Request Alle serverseitig verschlüsselten Daten verwenden das SQL Server 2000 Crypto API. Durch das Crypto API werden Funktionen zum Verwalten von Zertifikaten, Durchführen von Verschlüsselung und Entschlüsselung, zum Hashing und zur Erzeugung von Zufallszahlen bereitgestellt. Auf das API kann nur über nicht verwalteten Code zugegriffen werden. Dies sichert ein robustes und sicheres abspeichern von schützenswerten Inhalten des SQL Servers.

7.4.5 Rollenvergabe

Rollen erlauben es, Rechteinhaber in einzelne Einheiten einzuteilen, um sie einerseits besser verwalten und organisieren zu können und andererseits eine bessere übersicht über die Rechtevergabe zu bewahren. Der SQL Server 2000 verhält sich dabei identisch zur Rollenvergabe unter Windows.
Den zentralen Ansatz bilden dabei die Schlüsselworte:

- GRANT
- DENY
- REVOKE

Mit GRANT können Rechte vergeben, mit DENY verweigert und mit REVOKE bereits vergebene Rechte wieder aufgehoben werden. Die Menge aller Zugriffsrechte wird als Rolle in jeder Datenbank individuell definiert, wobei es auch datenbankübergreifende Rollen gibt. Der MS SQL Server 2000 realisiert DAC (s. Seite 3) komplett über Rollen, wobei das Augenmerk nicht nur auf der Einstufung von Benutzern und Gruppen liegt, sondern auch Datenbanken, Applikationen und sogar der ganze Server Rollen zugewiesen bekommen können.

Vergabe der Berechtigungen

Das Berechtigungssystem im SQL Server 2000 ist an das Berechtigungssystem von Windows angelehnt. Besitzt ein Benutzer mehrere Rollen, so erhält er die Vereinigungsmenge der Rechte dieser Rollen.

Ketten von Eigentumsrecht (Ownership Chains)

Das ,,Ownership-Chains-Konzept“ tritt in Kraft, sobald Berechtigungen bei Objekten überprüft werden. Ein denkbares Szenario wäre das überprüfen der Berechtigungen einer View. Zunächst wird der View und die entsprechenden Berechtigungen überprüft. Die Korrektheit der Berechtigungen der darüber liegenden Tabelle hingegen wird völlig außer Augen gelassen. Um ein solches Vorgehen zu vermeiden, gibt es Eigentumsrechte. Der SQL Server 2000 überprüft alle Berechtigungen, ob ein Bruch in der Benutzerkette der Eigentumsrechte vorliegt. Dieser liegt genau

dann vor, wenn beispielsweise ein View von Benutzer A, die zugrunde liegende Tabelle hingegen von Benutzer B erstellt wurde. Ein konstanter Benutzer der alle Objekte erstellt hat, ist somit nicht vorhanden. Nun kann es passieren, dass Benutzer A einem Benutzer C die Berechtigung für seinen View zuweist, Benutzer B damit aber nicht einverstanden ist und den Zugriff verweigert. Auf Grund dieser Verweigerung erhält Benutzer C keine Berechtigung auf den View zuzugreifen. Hätte Benutzer A den View, sowie die Tabelle entwickelt, so würden nur die Berechtigungen des Objekts View überprüft werden, da kein Bruch in der Kette vorliegt und Benutzer C hätte Zugang zum View.

7.4.6 Authentifizierung

Der SQL Server 2000 unterstützt zwei Arten der Authentifizierung. Einerseits die Windows- Authentifizierung und andererseits den so genannten Mixed Mode, welcher sowohl die Windows-, als auch eine DBMS interne Authentifizierung beinhaltet.
Die Windowsauthentifizierung gilt im Gegensatz zu Mixed Mode Authentication als ,,sicher“.
Während bei der Mixed Mode Authentication das Einrichten eines Benutzernamens und Passwortes für den Login zum SQL Server in der Mastertabelle ,,sysxlogins“ unumgänglich ist und diese Passwörter bereits geknackt werden können[16], benötigt der Nutzer des Windows Authentication Modes keine zusätzliche Kennung um Zugang zum SQL Server zu erlangen, da diese bereits bei der Windows Anmeldung erfolgt ist. Der Nachteil beider Varianten besteht in der Möglichkeit, dass das Passwort ,,blank“ gesetzt werden kann.
Obwohl der Mixed Mode nicht sicher ist, wird dieser noch immer verwendet, um eine Kompatibilität zu früheren SQL Server Versionen zu erreichen, die noch keine NTLM oder Kerberos Authentifizierungsprotokolle unterstützen.

7.4.7 Sicherer Serverzugang

Der Zugang zum Server wird von den zwei verschiedenen Arten der Authentifizierung [17] kontrolliert.

Zugang zum Server über die Windowsebene

Soll der Zugang auf Betriebssystemebene gesichert werden, muss der Administrator einen Login oder eine Anwendergruppe für die Nutzer erstellen, die Zugang zum DBMS erlangen wollen. Dies ermöglicht es alle Benutzer in einer Einheit zusammenfassen zu können und somit leichter wartbar zu machen. Der Administrator kann auf diese Weise kontrolliert Zugriffsrechte verteilen und entscheiden, welche Gruppe von Benutzern Zugang zum SQL Server 2000 erhält.

[16] beschrieben unter www.nextgenss.com/papers/cracking-sql-passwords.pdf
[17] s. 7.4.6 Seite 18

Zugang zum Server über die Serverebene

Auf der Serverebene muss den Gruppen, die unter Windows lokal erstellt wurden, das Recht für den Login zum SQL Server 2000 bereitgestellt werden.
Diese Zugriffsrechte können auch benutzerspezifisch und direkt erteilt werden. Diese Art der Rechtevergabe ist allerdings nur in sehr wenigen Situationen effektiv.[18]

7.4.8 Sicherer Datenbankzugang

Auf dem SQL Server 2000 bedeutet ein erfolgreiches Einloggen nicht automatisch den Zugang zu allen Datenbanken, die sich auf dem Server befinden. Zuvor müssen Zugangsberechtigungen für die einzelnen Datenbanken erfüllt sein. Hierfür ist es wichtig zu wissen, dass in jeder Datenbank der Benutzer mit einem SQL Server Login versehen wird und ein Windowsuser ist, oder aber einer Windowsgroup angehört. Der SQL Server Enterprise Manager sitzt auf der Microsoft Management Console (MMC). Hier wird eine Liste aller verfügbaren Konten mit Zugangsberechtigung zum SQL Server im Zusammenhang mit ihren zugehörigen Logins erstellt. Mit Hilfe dieser Liste kann nachvollzogen werden, welcher Benutzer welche Zugangsberechtigungen besitzt.

7.5 Effizienzbemerkungen

Discretionary Access Control und Mandatory Access Control bieten in ihren jeweiligen Implementationen sehr feingranulare und skalierbare Möglichkeiten zur Absicherung einer Datenbank gegen den nicht legitimen Zugriff durch Dritte, weisen jedoch bereits konzeptionell Fehler auf.

- So ist es beispielsweise einem Nutzer B möglich ein von Nutzer A erhaltenes Privileg nicht nur weiterzuverteilen, sondern auch wieder zu entfernen, so auch Nutzer A. Es ist also durch die transitiven Rechtebeziehungen möglich, dem Datenbankadministrator alle Rechte zu entziehen, die er zuvor einem anderen Nutzer erteilt hat.
- In keiner Weise wird durch die Verwendung von Security Policies das gefährliche SQL Injection verhindert
- Sollten in Sicherheitsmodulen wie Kerberos oder Windows Authentication Fehler bekannt werden, so sind auch automatisch Datenbanksysteme betroffen. Ein Update der entsprechenden Module dauert auf Grund der angepassten Implementierung dabei ungleich länger als das Update der Software selbst.

7.6 Ausblick

Mit jeder neuen Version, in der die Firmen Ihre DBMS veröffentlichen, wirken diese ausgereifter und sicherer, entfernen sich jedoch auch immer weiter von einem gemeinsamen Standard. Zwar unterstützen sowohl Oracle 11i als auch der SQL Server

[18] Beispielsweise bei einer sehr geringen Anzahl von Benutzern bzw. einer sehr kleinen Umgebungen.

2005 noch immer die Syntax nach SQL99, die Realisierung von MAC kann jedoch unterschiedlicher nicht sein, da diese nicht Teil des offiziellen SQL Standards ist. In aktuellen Versionen scheint jedoch die Weiterentwicklung von Trust Management und Policy Enforcement zu stagnieren. Stattdessen wird immer mehr Funktionalität in den DBMS Kern integriert. Microsoft erweitert die Funktionalität des SQL Servers um die .NET Strategie und die CLR(Common Language Runtime). ,,*Damit wird es künftig möglich sein Stored Procedures, Funktionen (UDFs) oder Triggers nicht nur in Microsofts SQL-Dialekt, T-SQL, sondern auch den .Net-Sprachen Visual Basic.Net oder C# zu schreiben[...]*“[19] Auch Oracle arbeitet vorrangig an der Funktionalität seines DBMS und bot mit seinem letzten Update für Oracle 11i nur kleinere Bugfixes, viele Neuerungen im Bereich E-Business, aber keine Neuerung im Bereich Security an![20]

7.7 Persönliches Fazit

Ein Vergleich beider vorgestellter DBMS gestaltete sich schwierig. Nicht nur, weil die betrachtete Oracle Version zwei Jahre jünger ist, als die aktuelle Version des SQL Servers, sondern auch, weil die anvisierten Zielgebiete beider Server auseinander gehen. Allein auf Basis der Menge an angebotenen Modulen betrachtet, ist Oracle momentan die bessere Entscheidung, wenn es um Datensicherheit geht. Die Menge an aktuellen Verschlüsselungsverfahren und die beinahe standardkonforme Implementierung von MAC wirken subjektiv reifer und weiterentwickelter als alles, was der SQL Server momentan zu bieten hat. Abhilfe gegen diesen technologischen Rückstand könnte nur Yukon schaffen, würde sich nicht genau hier betrachteten Teilbereich eine Stagnation in der Entwicklung abzeichnen.
Die scheinbare Stagnation der Entwicklung im Bereich Trust Management und Policy Enforcement scheint den Grund zu haben, dass die Entwickler Ihre DBMS momentan als sicher einstufen und vorrangig neue Funktionalitäten einbauen, um neue Märkte zu erschließen. Hier liegt aber auch die Gefahr. Je mehr Funktionalität in den Kern des DBMS integriert wird, desto größer ist die Wahrscheinlichkeit, dass sich in den immer komplexer werdenden Strukturen Fehler finden, die alle Sicherheitsvorkehrungen unwirksam machen können. Generell kann also gesagt werden, dass heutige DBMS sicher sind. Wie lange das jedoch so bleibt, bleibt abzuwarten.

[19] Entnommen von http://www.innovativetimes.ch/Default.aspx?tabid=77

[20] Eine Liste der in Oracle 11i hinzugefügten Features findet sich unter http://www.oracle.com/appsnet/technology/upgrade/docs/features.html

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Siemens AG. Siemens Online Lexikon. http://www.networks.siemens.de/solutionprovider/_online_lexikon/.

[2] Christian Bänsch. Relationale Datenbanken. Grundkurs Informatik - Bertha-v.-Suttner Gymnasium, 2000.

[3] Microsoft. MS Developer Network. http://msdn.microsoft.com/sqlserver/.

[4] Oracle. Oracle. http://www.oracle.com/.

[5] Oracle. *Oracle Advanced Security, Administrators Guide Release 2(9.2).* Oracle, 2 (9.2) edition, 2002.

[6] Oracle. *Oracle Label Based Security, Administrators Guide Release 2(9.2).* Oracle, 2 (9.2) edition, 2002.

[7] Shamkant B. Navathe Ramez Elmasri. *Grundlagen von Datenbanksystemen.* Addison Wesley, 3 edition, 2002.

[8] Reinhard Wobst. *Abenteuer Kryptologie - Methoden, Risiken und Nutzen der Datenverschlüsselung.* Addison Wesley, 3 edition, 2001.

8 Kerberos - Network Authentication Protocol

Wieland Rhenau, Christian Herrmann

8.1 Überblick über Kerberos und Rolle im Trust Management

Der folgende Abschnitt beschäftigt sich mit der Frage, was IT-Sicherheit eigentlich bedeutet und welche Rolle Software wie Kerberos in diesem Umfeld spielt.

8.1.1 IT-Sicherheit Heute und ihre Begriffswelt

Wir leben in einer Zeit, in der die rasante Ausbreitung der globalen Vernetzung und immer komplexere IT-Infrastrukturen Mechanismen erforderlich machen, die Kontroll- und Sicherheitsanforderungen gerecht werden, aber auch die Benutzerfreundlichkeit nicht vernachlässigen. Die interne Sicherheit von Netzen hat dabei, genauso wie die Abschirmung nach Außen Priorität, da der Zugriff auf sensible Daten nur authorisierten Personen gestattet werden darf, um der zunehmenden ökonomischen Bedeutung von Information nachzukommen. Grundsätzlich ist zu sagen, das Verbindungen generell unsicher sind. Sie können unterbrochen aber auch umgeleitet werden. Diese Folgen, ebenso wie das Abhören oder Auslesen bilden Zugriffsmöglichkeiten auf Daten, die streng vertraulich sind. Dabei ist auch die private Kommunikation nicht außer Acht zu lassen. Da die Privatsphäre als Gut und vor allem als Grundrecht eines jeden Menschen in der heutigen Zeit stets bedroht ist. Aufgabe von Sicherheitsmechanismen muss demnach sein, insbesondere die Übermittlung von hoch relevanten Daten wie Passwörtern besonders zu überwachen und zu reduzieren. Ein wesentliches Konzept dies zu realisieren ist das in Kerberos umgesetzte Single-Sign-On Konzept, welches Authentifikation und Autorisierung trennt. Dieses Prinzip wird im Folgenden näher beschrieben. Verwendete Protokolle zur Kommunikation wie http, FTP, POP oder Telnet bieten Angriffsmöglichkeiten, denen man durch den zusätzlichen Einsatz von Integritätssicherungs- und Verschlüsselungsverfahren entgegen wirken muss um auch die Sicherheit und Echtheit der transportierten Daten zu gewährleisten. Wesentliche Hauptursachen von Sicherheitsproblemen sind konzeptionelle Mängel im Software-Design und Implementierungsfehler sowie administrative Fehler, denen es - wenn vorhanden entgegen zu wirken gilt. Um über IT-Sicherheit sprechen zu können muss man sich über grundsätzliche Begriffe und deren Tragweite in Klaren sein, die im Anschluss erläutert werden.

Objekt:

Als (Daten) Objekt bezeichnet man Dateien, Programme und Prozesse. Dabei unterscheidet man zwischen passiven und aktiven Objekten. Passive Objekte wie Dateien oder Programme sind in der Lage Informationen zu speichern, wobei aktive Objekte wie Prozesse zusätzlich die Eigenschaft haben, Informationen zu lesen und zu verändern.

Subjekt:
Über Schnittstellen (Operationen, Methoden) können andere Objekte (Server, Prozeduren) oder Benutzer auf Objekte zugreifen und diese verarbeiten. Benutzer und Objekte, die innerhalb eines Systems Zugriff auf die Okjete haben, werden als Subjekte bezeichnet.

Spionage:
Unberechtigte Kenntnisnahme von Objekten

Sabotage:
Unberechtigte Änderung und Missbrauch von Objekten

Echtheit:
Als Echtheit bezeichnet man die Glaubwürdigkeit eines Objekts oder Subjekts, welche anhand seiner Identität und seiner charakterisierenden Eigenschaften überprüfbar ist.

Integrität:
(Daten)Integrität ist gegeben, wenn sichergestellt ist, das es Subjekten nicht möglich ist, die zu schützenden Daten unautorisiert und unbemerkt zu manipulieren.

Robustheit:
Robustheit ist die Eigenschaft eines Systems, seltenere Vorkommnisse der Umwelt, die in ihren Eigenschaften stark abweichend sind, geschehen lassen zu können, ohne dass der Fortgang des Systems hiervon wesentlich betroffen ist.
Quelle: http://www.elexi.de/de/r/ro/robustheit.html

Kryptographie:
Unter Kryptographie versteht man die Lehre der Ver- und Entschlüsselung von Daten. Dabei wird erreicht,dass einem unbefugten Benutzer die Semantik der Information vorenthalten wird. Zusätzlich hat man die Möglichkeit zu erkennen, ob Daten verändert worden sind.

Denial-of-Service:
Diese spezifische Angriffsklasse - auch als Resource Clogging bezeichnet - umfasst Attacken, die auf die Verfügbarkeit von Systemkomponenten oder -diensten abzielen und diese durch zum Beispiel durch Überschwemmen von Netzen mit Nachrichten angreifen und Dienste lahm legen und den Netzverkehr beeinträchtigen .

Authentifizierung und Autorisierung werden im nächsten Abschnitt erläutert.

8.1.2 Motivation für Authentifikation und Authorisierung

Um Sicherheit im Umgang mit Informationen zu gewährleisten ist es erforderlich, Benutzer eindeutig zu identifizieren. Dieser Prozess wird als Authentifizierung bezeichnet. Des Weiteren werden den Benutzern individuell festgelegte Zugriffsrechte zugewiesen, was Autorisierung genannt wird. Damit wird sichergestellt, dass nach der Verifikation der Identität nur autorisierte Aktionen auf bestimmte Informationen möglich sind.

8.1.3 Kerberos der Mythos

In der Informatik werden oft Bezeichnungen für Implementierungen oder Komponenten gewählt, die einen Zusammenhang zwischen Namen und Funktionalität repräsentieren. Kerberos als Name hat seinen Ursprung in der griechischen Mythologie und symbolisiert einen dreiköpfigen Höllenhund, der die Unterwelt bewacht. Er gewährte den Zugang, jedoch ließ er niemanden aus seinem Reich zurückkehren. Vielleicht symbolisiert dies die Notwendigkeit der Zugriffskontrolle (es geht nicht mehr ohne). Jeder der drei Köpfe stellt eine der Kerberos Komponenten (Client, Server, Key Distribution Center)dar. Aus diesem Grund ist der dreiköpfige Hund auch das Logo von Kerberos.

8.1.4 Entstehung von Kerberos

In seinem Urprung war Kerberos gar nicht als separate Implementierung angedacht.Im Jahr 1983 wurde am Massachusetts Institute of Technology (MIT) am ATHENA- Projekt gearbeitet, welches als Softwarerealisierung einer Client-Server-Infrastruktur für graphische Workstations am universitätsinternen Campusnetz gedacht war. Dieses umfasste 25000 Nutzer und 1200 Computer. Ein wesentlicher Aspekt des Projekts war eine Komponente, die es ermöglichen sollte, ein zentrales Authentifikationssystem zu entwickeln. Daraus ging Kerberos hervor, das erstmals 1986 eingesetzt wurde. Die Grundlage war ein von Needham-Schroeder entworfenes Authentifikations-Protokoll. Dies wurde durch S.Miller und C.Neuman weiterentwickelt und um die Verwendung von Zeitstempeln ergänzt, was von Denning und Sacco vorgeschlagen wurde. Daraus resultierte die Erstimplementation von Kerberos. Bis 1987/88 wurde Kerberos MIT-intern bis hin zur Version 3 weiterentwickelt . Mit der ersten frei verfügbaren Version 4 beteiligte sich neben DEC auch IBM an der Entwicklung des Projekts. In der Folgezeit wurde Kerberos bis 1990 weiter vebessert aber schon 1989 begannen die Entwickler mit der Spezifikation zur Version 5, die einige grundsetzliche Mängel beheben sollte. Im September 1993 wurde die Spezifikation zum international anerkannten Standard als RFC 1510. Mittlerweile ist Kerberos in großer Vielfalt von freien und kommerziellen Distributionen erhältlich. Es ist fester Bestandteil von Betriessystemen wie Windows, SunOS, Solaris, Linux und weiteren Unix-Derivaten. Da die USA im Bereich digitaler Verschlüsselungstechnologien strenge Richtlinien verfolgen, die den Export extrem einschränken entstand als Pendant zur Kerberos Version 4 eine freiverfügbare Implementation mit der Bezeichnung "BONES", auf die jedoch nicht näher eingegangen wird. Die entstandenen Probleme beim Export von Kerberos

wurden allerdings mit der Version 5 behoben, da nun das Kryptographie-Verfahren als Modul innerhalb von Kerberos frei wählbar (austauschbar) ist.

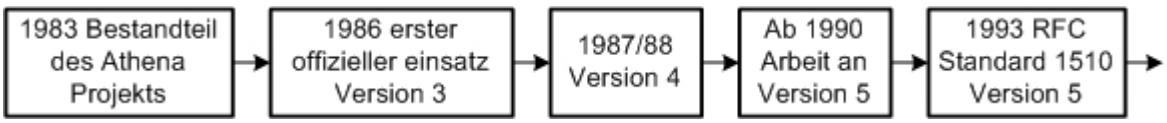


Abbildung 8.1: Entstehung von Kerberos

8.1.5 Grundsätzliche Architektur

Um die grundsätzliche Architektur von Kerberos zu beschreiben, wird folgend der Aufgabenbereich von Kerberos eingegrenzt.

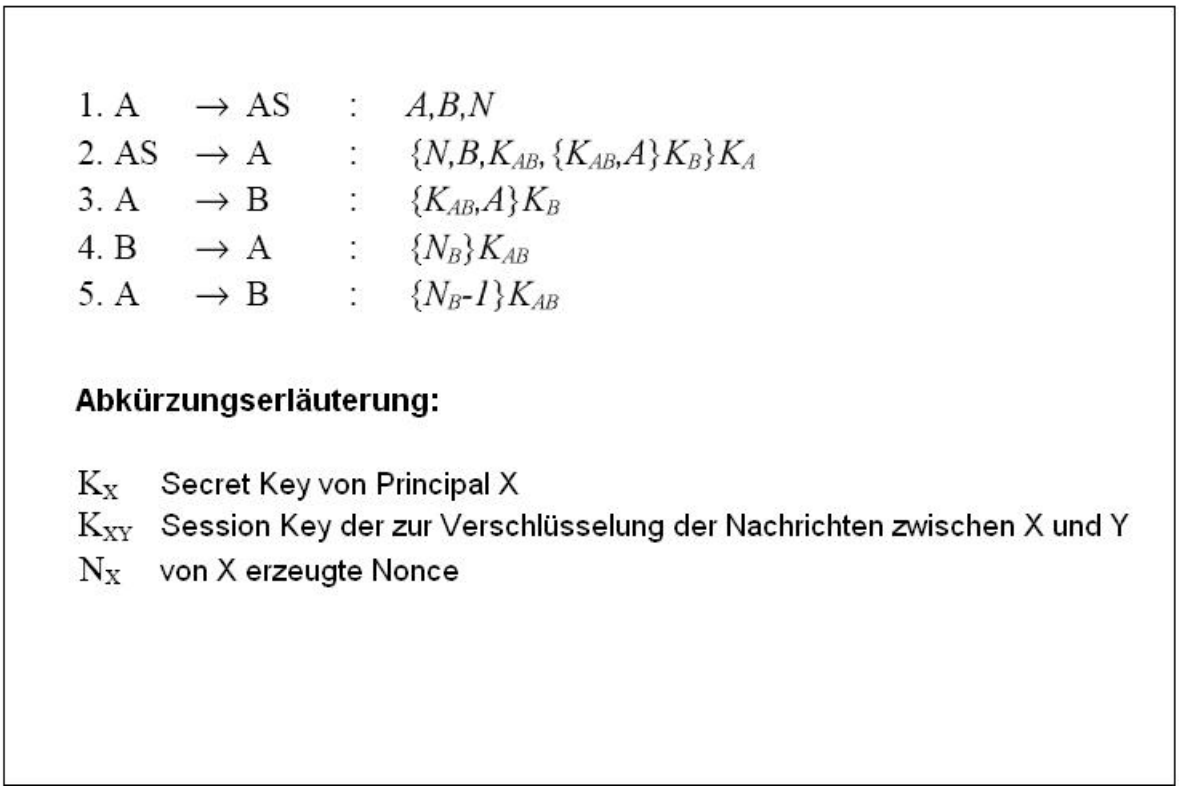
Die Implementation des Kerberos Systems in ein Kommunikationsnetz erfolgt zumeist auf der Ebene der Anwendungsschicht, bezogen auf die Schichten des ISO-OSI Referenzmodells somit auf der End-zu-End-Ebene 7. Die End-zu-End Verbindungsbeziehung der Kommunikationsinstanzen bedingt eine diesbezügliche Authentifikationssicherheit, da auf unteren Ebenen Sicherheits- maßnahmen nicht ausreichend sind. Eine Implementation auf ISO-OSI Ebene 6 z.B. mit RPC, oder auch die Integration auf tieferen Protokollebenen von IP, TCP oder UDP ist mit Nutzen für die Host-zu-Host Sicherheit möglich, findet aber selten Anwendung.

Kerberos ist eine Client-Server-Applikation, die eine Authentifizierung mittels Tickets realisiert. Authentifikaztionsinformationen werden in Kerberos dezentral verwaltet, wobei die Basis eine festgelegte Hierachie von Authentifikations-Servern ist. Die Server haben autonome Verantwortungsbereiche, was Bereichsübergreifende Zugriffe von Benutzern transparent umsetzt. Als wichtige Eigenschaft von Kerberos ist die Umsetzung des Single-Sign-On Konzepts zu nennen. Dieses bedeutet, das der Benutzer sich lediglich einmal authentifizieren muss, um auf verschiedene Dienste und/oder Daten zugreifen zu können. Dies verringert den Bedarf, Passwörter unnötig oft zu versenden. Kerberos basiert auf symmetrischen Verschlüsselungsverfahren (zum Beispiel DES), wobei es davon ausgeht, dass beide Principals, zwischen denen die Kommunikation stattfinden soll, ihre Schlüssel sicher verwalten. Erweiterungvorschläge einer asymmetrischen Verschlüsselung sind angedacht jedoch noch nicht umgesetzt. Die Mitgliederverwaltung innerhalb von Kerberos findet in einer Datenbank statt, auf die nur der Server Zugriff hat. Die Authentifizierung in Kerberos findet lediglich indirekt statt. Das heißt die Erlaubnis zur Kommunikation wird durch ein Ticket gestattet, wobei die Session der Kommunikation zeitbegrenzt ist.

Die zwei ursprünglich für Kerberos verwendeten Technologien sind das Needham Schroeder Protokoll und die Zeitstempel.

Needham Schroeder Protokoll:
Die Grundlage vieler in der Praxis eingesetzten Protokolle sind Basisprotokolle

zum Schlüsselaustausch wie die Needham Schroeder Protokolle. R. Needham und M. Schroeder haben diese bereits 1978 entwickelt.



Das gemeinsame Geheimnis, das nur Benutzer und die Trusted Third Party kennen ist im Needham Schroeder Protokoll ein Secret Key. In der folgende Erläuterung wird die Trusted Third Party als Authentication Service (AS) bezeichnet. Die Authentifikation eines Principal A zu einem Principal B wird in der Abbildung 1.2 erläutert.

Den aktiven Part übernimmt A und ist sozusagen der Client, wobei B in der passiven Rolle des Servers ist. Zu Beginn schickt A in der ersten Nachricht seinen Namen und den Namen des erwünschten Kommunikationspartners mit einer Nonce an den AS. (Nonce zeit varianter Parameter also Counter oder Zeitstempel)Der Authorisierungsdienst überprüft zuerst, ob die Telnehmer A und B bei ihm registriert sind. Wenn dies so ist, sendet er als Antwort die Nachricht 2. Um sicherzustellen ,dass nur A die Antwort lesen kann, ist sie mit KA verschlüsselt. Ihr Inhalt besteht aus dem Sessions Key KAB, der Grundlage zur Verschlüsselung von Nachricht 4 und 5 ist, der Nonce aus Nachricht 1 zur Replay-Erkennung, dem Namen von B und einen Teil der mit KB, der den Session Key KAB und den Namen von A beinhaltet. Wenn A widerum die äußere Verschlüsselung decodieren kann, ist A authentifiziert. Die Authentifikation von AS ist auch gegeben, da außer A ja nur AS mit KA verschlüsseln kann. Ist bis zum jetzigen Zeitpunkt alles ohne Probleme verlaufen, sendet A in der Nachricht 3 den durch KB verschlüsselten

Teil an B. Wenn B diese Nachricht decodieren kann, ist auch B authentisch. Da B allerdings einen Beweis für die Authentität von A haben möchte, schickt B in Nachricht 4 eine Nonce zu A die mit KAB chiffriert ist. Die Beantwortung von Nachricht 4 durch A authentifiziert A für B. Diese Antwort ist Nachricht 5. Somit haben sich A und B gegenseitig als vertrauenswürdig eingestuft, ohne das Versenden eines geheimen Schlüssels untereinander. AS hat in Nachricht 2 Informationen an A geschickt, die A nicht entschlüsseln und lediglich an B weiterreicht. Dies ist ein sicherer Schlüsselaustausch, der in Kerberos Anwendung findet. In Kerberos wird dieser fremdverschlüsselte Part als Ticket bezeichnet. Mittels dieses Tickets sendet A den Session Key an B, ohne dass die Angreifer ihn lesen können. Ohne Modifikation ist das Needham Schroeder Protokoll aber nicht in Kerberos einsetzbar, da es zwei wesentliche Einschränkungen besitzt. Als erstes ist es nicht möglich, dass A nebenläufig mehrere B´s authentifiziert, da A nicht herleiten kann von welchem B die benötigte Nachricht 4 stammt. Auch B kann nicht nebenläufig mehrere A´s identifizieren, da B das gleiche Problem durch Nachricht 5 hat wie A. Hat man jedoch Netze mit vielen Principals ist dieser Nachteil äußerst unpraktisch. Das zweite Problem besteht darin, das ein Angreifer offline versuchen kann, den Session Key mittels Nachricht 4 und Nachricht 5 zu decodieren. Wenn dies geschafft ist, kann der Angreifer die abgehörte Nachricht 3 verschicken und sich als A ausgeben. Dies könnte man nur verhindern, indem jeder Session Key nur ein einziges Mal verwendet wird und B sich alle verwendeten Session Keys merkt. Session Keys zu speichern ist aber genauso unpraktikabel wie das erste Problem. Diesen Gedanken haben wie bereits erwähnt, Denning und Sacco aufgegriffen und das Needham Schroeder Protokoll um die Zeitstempel erweitert. Diese bewirken, dass sich B nur Session Keys aus einem bestimmten Zeitintervall merken muss was den Nachteil auf das erste Problem beschränkt.

Zeitstempel:
Das vorgestellte Authentifikationsprotokoll ist grundsätzlich sicher, wenn man die Möglichkeit ausschliesst, dass kein geheimer Schlüssel in die Hände eines Eindringlings fällt. Davon ist aber nicht in jedem Fall auszugehen und wenn ein Eindringling genügend Zeit hat, könnte er diesen auch entschlüsseln. Nun könnte er eine von ihm aufgezeichnete Nachricht von sich aus erneut versenden und der angesprochene Teilnehmer würde glauben, dass der original Absender eine Nachricht schickt. Um diesem Problem aus dem Weg zu gehen, versieht man Nachrichten mit einer Uhrzeit. Der Empfänger muss die angegebene Zeit mit seiner Systemzeit vergleichen und anhand einer vereinbarten Toleranz entscheiden, ob die Nachricht gültig ist. Dabei wird bei symmetrischen Protokollverfahren davon ausgegangen, dass alle Teilnehmer über synchronisierte Systemzeiten verfügen.

Kerberos allgemein:
Die erste der zwei elementaren Aufgaben von Kerberos ist Authentifikation von Subjekten (Arbeitsplatzrechner, Server oder Benutzer). Diese Subjekte heißen in Kerberos Principals. Als zweite Hauptfunktion ist der Austausch von Sitzungsschlüsseln in Kerberos zu bezeichnen. Die Basis vieler Authentifikationsverfahren bildet der Einsatz eines Vermittlers zwischen den Principals. Dieser ist vertrauenswürdig und wird als Trusted Third Party bezeichnet. Auch Kerberos verwen-

det diesen, wobei jeder Principal ein geheimen Schlüssel besitzt, den nur er und der Vermittler kennt. Somit können sich der Pricipal, der kommunizieren will und der Vermittler gegenseitig identifizieren. Dieser Prozess wird im Abschnitt 2 näher erläutert.

8.2 Komponenten

In nächsten Abschnitt soll die Funktionsweise von Kerberos näher erläutert werden. Da es einige wichtige Komponenten gibt, die in der Kommunikation eine große Rolle spielen, werden wir in diesem Artikel die Grundlagen dafür vermitteln.

Prinzipals:
Jeder, der das sichere Kerberos-Netzwerk benutzt, wird als Prinzipal bezeichnet. Dabei gibt es kleine Unterschiede in der Darstellung zwischen den Prinzipals im Kerberos 4 und Kerberos 5. Bei Kerberos 4 sieht die Darstellung wie folgt aus:
name.instance@realm
Der Kerberos 5 Prinzipal hat hingegen den folgenden Aufbau:
component/component/component@realm

ACL:
Auf dem Kerberos Server werden alle Prinzipals zusammen mit ihrem Passwort abgespeichert. Die Liste in der sie eingetragen sind, wird "Access Controll List" genannt.

AS:
Die Abkürzung AS steht für den Authentication Service. Welche Rolle dieser Service bei Kerberos spielt, soll erst im Kapitel "Funktionsweise" näher erläutert werden.

TGT:
Um während der Arbeit in einem Kerberos gesicherten Netzwerk auf den Großteil der Passwortversendungen verzichten zu können, wurde ein "Ticket Granting Ticket" eingeführt. Dieses Ticket gilt in der laufenden Kommunikation als Passwort Ersatz.

TGS:
TGS steht für "Ticket Granting Service". Er verifiziert das "Ticket Granting Ticket" des Prinzipals und übergibt ein Ticket mit dem der Prinzipal weitere Dienste nutzen kann.

KDC:
Auf dem Kerberos Server befindet sich das KDC. Also das "Key Distribution Center". Es verwaltet die Komponenten Äuthentication Service", "Ticket Granting Serviceünd Äccess Controll List".

Realm:
Das deutsche Wort für Realm ist "das Reich". Damit wird der gesamte Netzwerkto-

pologische Bereich definiert, in dem sich die Prinzipals eines Kerberos Servers befinden.

PAM:
PAM steht für "Plugable Authentication Modules" und bedeutet genau das, was es übersetzt heißt. Es ermöglicht dynamisch Module zur Authentifikation einzusetzen.

8.3 Funktionsweise

Im letzten Kapitel wurden die wichtigsten Komponenten eines Kerberosnetzwerkes vorgestellt. Dieses Kapitel soll als Grundlage dienen, um die mehr oder minder komplexe Funktionsweise eines Kerberosnetzwerkes zu verstehen. Der Einfachheit halber und um Verwechslungen zu vermeiden gehen wir im Folgenden davon aus, dass unsere Kommunikation in einem von Kerberos 5 geschützten Netzwerk stattfindet.

An dieser Stelle soll noch einmal in Erinnerung gerufen werden, dass der Kerberos Server das KDC und somit auch die drei Komponenten AS ("Authentication Service"), TGS ("Ticket Granting Service") und ACL ("Access Controll List") beherbergt. Bevor ein Prinzipal das Kerberosnetzwerk und seine Services das erste Mal benutzen kann, muss er vom Administrator in die "Access Controll List" des Kerberos Servers eingetragen werden. Dafür wird sowohl der Name des Prinzipals als auch das zugehörige Passwort der Liste hinzugefügt. Des Weiteren wird dem Server mitgeteilt, welche Dienste der Prinzipal in dem Netzwerk benutzen darf. Die Grundlage für das sichere Arbeiten ist geschaffen worden. Jedes Mal wenn der Prinzipal jetzt zu diesem Netzwerk hinzu stößt, muss er sich beim Kerberos Server, genauer beim "Authentication Service", authentifizieren lassen. Diese Authentifizierung kann auf zwei unterschiedlichen Wegen geschehen. Entweder gibt der Prinzipal den Befehl *kinit* und sein Passwort ein, oder der Befehl wird automatisch mit dem Login verknüpft. An Hand des Benutzernamens und des Passwortes kann der AS erkennen ob der Prinzipal in der ACL steht. Ist dies der Fall, so erhält der Prinzipal ein TGT ("Ticket Granting Ticket") mit dessen Hilfe er sich jeder Zeit im KDC Service-Tickets zur Kommunikation mit anderen Prinzipals besorgen kann. Das TGT gilt in der fortlaufenden Kommunikation als Passwortersatz und ist nur eine begrenzte Zeit gültig. Diese Zeitbegrenzung wurde bei der Kerberoskonfiguration gesetzt. Je kürzer die Zeitspanne, desto sicherer ist das Netzwerk. Jedoch sinkt der Komfort, umso häufiger ein TGT beim Kerberos Server nachgefordert werden muss. Hier muss also ein guter Kompromiss geschaffen werden. Da die fortlaufende Authentication mit dem TGT stattfindet, ist das Passwort relativ sicher. Gelingt es jemandem, das TGT abzufangen, kann er sich nur für die restliche Gültigkeitsdauer des Tickets als der falsche Prinzipal ausgeben. An dieser Stelle stellt sich jedoch die Frage, was man mit diesem TGT anstellen kann. Alles! Jedenfalls alles was dem Prinzipal bei der Einrichtung erlaubt wurde. Um einen Service zu nutzen, geschieht folgendes. Der Prinzipal möchte eine Telnet-Verbindung zu einem anderen Prinzipal aufbauen. Dafür benutzt er den Befehl *telnet -a -x lisa.cs.uni-potsdam.de*. Durch das Attribut *-a* wird

das TGT zur Anmeldung benutzt. Es entfällt daher die Eingabe des Passwortes. Das Attribut -*x* gibt die Anweisung zur Verschlüsselung der Kommunikation. Der Verschlüsselungsalgorithmus war standardisiert auf DES. Durch ein Plugin-Service in Kerberos 5 ist es jedoch gestattet die relativ schwache Verschlüsselung DES durch eine stärkere wie 3DES etc. auszutauschen. Hinter den Kulissen geschieht folgendes. Der Prinzipal ist noch nicht im Besitz eines Service-Tickets, das ihm die Verbindung zu dem Prinzipal ermöglicht. Daher wird automatisch, mit Hilfe seines TGT, beim TGS ein Service-Ticket für diesen Dienst angefordert. Das Service Ticket trägt das Verfallsdatum des TGT. Dadurch wird auch hier ein hohes Maß an Sicherheit erreicht und der Prinzipal kann alle Dienste die ihm zustehen, sicher im Netzwerk benutzen. Beim gegenüberliegenden Prinzipal reicht er jetzt eine Kopie des Service-Tickets ein und die Telnet-Verbindung steht.

8.4 Praktisches Anwendungsgebiet

Das folgende Kapitel soll einige der praktischen Anwendungsgebiete von Kerberos vorstellen.

In erster Linie sind hier die Kerberos angepassten Applikationen zu nennen. Dazu gehört vor allem die weit verbreitete und bereits erwähnte Applikation telnet und die nicht so bekannten r-Tools. Zu den r-Tools gehören rlogin, rcp und rsh. Die Beschreibung dieser Tools wird nicht nötig sein und somit hier auch ausgelassen. Letztendlich bleiben noch die Kerberostauglichen Dateisysteme zu nennen. Dazu gehören NFS und AFS. Die meisten Distributionen unterstützen Kerberos. Häufig befindet sich der Quellcode auf der mitgelieferten CD/DVD und kann bei Bedarf einfach nachinstalliert werden.

8.5 Entwicklung bis heute

Der größte Sprung in der Entwicklung von Kerberos wurde beim Übergang von Kerberos 4 auf Kerberos 5 vorgenommen. Da die Veränderungen zahlreich sind, werden nachfolgend die wichtigsten aufgeführt.

In erster Linie wurde der Key Salt Algorithmus geändert. Bei diesem Algorithmus handelt es sich um eine Verbesserung der Verschlüsselung. Das Key Salt ist, wie der Name schon sagt, das Salz in der Verschlüsselung. Da es immer noch User gibt, die sich besonders einfache Passwörter ausdenken, wird jedes Passwort durch einen Algorithmus und einen erzeugten String verkompliziert.

Des Weiteren wurde das Netzwerk-Protokoll komplett überarbeitet. Kerberos 5 benutzt ASN.1. ASN.1 gilt als Abstrakte plattformunabhängige Beschreibungssprache, mit ihr kann man festlegen, wie aus einfachen Datentypen komplexe Datentypen zusammengebaut werden.

Die Funktionalität des Tickets wurde erweitert. Durch das Setzen einiger bestimmter Bits, erhält ein Ticket eines der folgenden Attribute. Entweder "forwardable",

"rewardable" oder "postdatable". Ist das Bit für "forwardable" gesetzt, dann kann das Ticket weitergeleitet werden. Falls das Bit für "renewable" gesetzt ist, kann das Ticket wieder erneuert werden. Für den Fall, dass das Bit für "postdateable" gesetzt ist, kann das Ticket auf einen späteren Zeitpunkt datiert werden.

Seit Kerberos 5 ist es möglich, dass Tickets sowohl multiple IP-Adressen als auch Adressen unterschiedlichen Netzwerkprotokolltyps beinhalten.

Ein Verschlüsselungs-Interface Modul wird benutzt und ermöglicht somit den Einsatz verschiedener Verschlüsselungsalgorithmen. Bis dato stand lediglich DES zur Verfügung. Da DES nicht mehr als besonders sicher gilt, kann bei Kerberos 5 durchaus auf 3DES zurückgegriffen werden.

Wie bereits erwähnt, werden Kerberos-Tickets nach Ablauf einer bestimmten Zeit ungültig. Diese Eigenschaft des Tickets verleiht Kerberos ein gewisses Maß an Sicherheit. Jeder der illegal in den Besitz des Tickets eines anderen Prinzipals gelangt, hat somit nur bis zum Ablauf des Tickets Zugriff auf den entsprechenden Service. Allerdings ist das immer noch ein Risiko, welches für viele nicht tragbar ist. Aus diesem Grund gibt es seit Kerberos 5 den Replay-Cache. Er verhindert das Tickets mehr als einmal Verwendung finden. Jeder Prinzipal der sich sein Ticket erst besorgt, wenn er es auch wirklich benötigt, muss den Missbrauch nicht fürchten.

Kerberos Prinzipals sind in der Lage sich gegenüber anderen Prinzipals aus dem eigenen Realm zu authentifizieren. Kerberos-Realms können auch so konfiguriert werden, dass sich Prinzipals aus dem einen Realm gegenüber Prinzipals aus dem anderen Realm authentifizieren können. Seit Kerberos 5 gibt es Transitive Cross-Realm-Authentication. Dafür wird ein Pfad definiert, durch den man springen kann, bis man beim gewünschten Realm angelangt ist.

Bei Kerberos 5 spielt Preauthentication eine besondere Rolle. Preauthentication bedeutet, dass der Prinzipal sich zusätzlich beim KDC authentifiziert um ein TGT zu erhalten. Die einfachste Form der Preauthentication läuft unter dem Namen PA-ENC-TIMESTAMP und ist der aktuelle Zeitstempel der mit dem Schlüssel des Prinzipals verschlüsselt wird.

8.6 Systemschwächen

Nachdem Kerberos ausführlich beschrieben wurde, soll nun auf die Schwachpunkte der Architektur eingegangen werden. Keine Sicherheitspolitik kann das Prädikat "PERFEKT"für sich in Anspruch nehmen und so bietet auch Kerberos Angrifffspunkte und Probleme bei der Zusammenarbeit unterschiedlicher Versionen. Als Hauptschwäche ist zu nennen, dass Kerberos sich nicht eignet für zwei aktiv miteinander kommunizierende Instanzen, wie zwei menschlichen Benutzern. Die Kommunikation erfolgt immer nur zwischen einem aktiv agierenden Client (Benutzer) und einem passiv reaktiven Server.

8.6.1 Versionsinkompatiblität

Mit dem Schritt von Kerberos 4 zu Kerberos 5 treten neben den positiven Neuerungen allerdings Probleme auf, die es zu berücksichtigen gilt. Version 5 ist zwar abwärtiskompatibel zu Version 4 aber anders herum funktioniert der Versoinsübergang nicht.

8.6.2 Angriffsmöglichkeiten

Kerberos ist prinzipiell eine Alles-oder-Nichts-Lösung. Verwendet man Kerberos im Netzwerk, müssen sich die Benutzer alle Passwörter merken, die an einen Dienst übertragen werden, der nicht kerberisiert ist. Zusätzlich existiert die Gefahr, dass diese Passwörter von Packet Sniffern erfasst werden. Dies bedeutet, es ergibt sich für Ihr Netzwerk keinerlei Vorteil aus der Verwendung von Kerberos wenn man Anwendungen benutzt, die nicht durch Kerberos gesichert sind. Ein sicherer Schutz ist nur gegeben, wenn man entweder alle Anwendungen, die Passwörter im Klartext versenden, kerberisiert, oder man auf die Verwendung dieser Anwendungen im Netzwerk verzichtet. In Kerberos Version 4 kann Jeder verschlüsselte Credentials eines beliebigen Benutzers anfragen, da beim AS_REQ lediglich eine Benutzerkennung versendet werden muss. Ein weiterer Nachteil ist, dass durch die in Version 5 mögliche Preauthentification die clienteigene Verwendung des Passwortes verlängert wird. Dies hilft insbesondere bei der Offline Deschiffrierung der Keys. Der geheime Schlüssel des angefragten Services ist beim Benutzer immer in Form des mit ihm verschlüsselten Tickets vorhanden. Das gleiche gilt für den Master Key des KDCs, mit dem von ihm verschlüsselten TGT. Auch hier könnte eine Offline-Entschlüsselung eingesetzt werden. Abschliessend ist zu sagen, dass Kerberos keinen Schutz vor Trojanern oder anderen möglich Schnüfflern gewährt, dazu zählen natürlich auch nichtauthorisierte Benutzer. Die Identifikation eines Teilnehmers erfolgt bei Kerberos vollständig über sein geheimes Passwort. Das bedeutet, je schlechter dieses Passwort ausgewählt wird, um so schlechter ist der Schutz -selbst bei guten Verschlüsselungsverfahren. Sinnvoll wäre die Kombination des Passwortes mit zusätzlichen Identifikationsmerkmalen, wie etwa Smart-Cards oder Biometrischen Daten.

8.7 Zusammenfassung

Nach der Analyse von Kerberos und den Verbesserungen zur Version 5 kann man sagen, dass bis auf die aufgezeigten Schwächen Kerberos grundsätzlich als sehr empfehlenswert einzustufen ist. Durch sichere Kommunikation, Secret Key Verfahren und Datenverifikation, sind Inhalte ebnso wie Nutzer relativ gut geschützt vor Angreifern. Sicherlich ist Kerberos keine perfekte Applikation, aber im Bereich Applikationssicherheit bietet es im Hinblick auf Benutzerfreundlichkeit im ausgewogenen Verhältnis zur Sicherheit ein durchdachtes Konzept. Die vielseitige Verwendung von Kerberos in diversen Betriebssystemen bezeugt diese Aussage. Zur Kommunikation im Internet - also in einem Wide Area Network ist Kerberos sicherlich nicht geeignet. Autonomen Netzen bietet es jedoch ausreichende Sicherheit, wenn durch andere Vorkehrungen Host-Sicherheit gegeben ist und die Nutzer

nicht zu einfache Passwörter wählen.

Literaturverzeichnis

[1] C. Eckert. *IT-Sicherheit*. Oldenbourg Wissenschaftsverlag GmbH, 1 edition, 2001.

[2] Everyone. Kerberos (Informatik). http://de.wikipedia.org/wiki/Kerberos_%28Informatik%29, Aug 2000.

[3] Ken Hornstein. Frequently Asked Questions about the Kerberos. http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html, 2004.

[4] MIT. Kerberos: The Network Authentication Protocol. http://web.mit.edu/kerberos/www/, Jun 2004.

[5] Inc. Red Hat. Kerberos. http://www.europe.redhat.com/documentation/rhl7.2/rhl-cg-de-7.2/kerberos.php3, 2004.

[6] Peter Wächtler. Eine Frage des Vertrauens. http://www.linux-magazin.de/Artikel/ausgabe/1999/05/Kerberos/kerberos.html, Mai 1999.

9 Intersystemoperierende RBAC-Implementierungen

Manuel Dräger, Tobias Grave

Abstract

RBAC ist eine Zugriffskontrollstrategie insbesondere für große IT Systeme mit vielen zu verwaltenden Benutzern, es vereinfacht durch einen rollenbasierten Ansatz die Sicherheitsadministration erheblich und ermöglicht die Umsezung verschiedener Sicherheitspolitiken. Durch die Zeitersparnis bei der Administration und dem Gewinn von mehr Sicherheit durch eng an der Organisationshierarchie orientierten Zugriffsberechtigungen kann ein RBAC-System Kosten sparen und Sicherheitsprobleme vermeiden helfen.

Das folgende Dokument ist auch online unter

http://burns.cs.uni-potsdam.de/~mdraeger/

als LaTeX- bzw. PDF-Version erhältlich.

9.1 Einleitung

In vernetzten Rechenumgebungen mit vielen Benutzern und bei verteilten Anwendungen ist Zugriffskontrolle unverzichtbar, um sensible Daten vor Missbrauch zu schützen. Diese Daten können Dateien, Speicherobjekte, Prozesse oder Prozeduren sein, im folgenden werden sie allgemein als Objekte bezeichnet. Die Zugriffsrechte auf diese Objekten sind lesen, schreiben, ändern und ggf. ausführen. Ein Zugriffskontrollsystem kann anhand dieser Rechte für jeden Benutzer entscheiden, ob er die nötigen Rechte für eine gewünschte Aktion hat und sie dann zulassen bzw. verweigern.
Problematisch ist, dass die Verwaltung von Rechten in großen, verteilten IT Systemen sehr aufwändig und fehleranfällig ist, etwa wenn im Extremfall die Rechte manuell für jeden einzelnen Rechner eingegeben werden müssen. Neben dem hohen Zeitaufwand bei Einrichtung und Pflege der Systeme ist auch das Fehlerrisiko sehr hoch, da sich schnell kleine Flüchtigkeitsfehler einstellen.
Eine zentrale Rechteverwaltung kann die Administration stark vereinfachen und das Fehlerrisiko minimieren, doch nicht immer ist mit Bordmitteln der Betriebssysteme oder Anwendungsserver die Durchsetzung einer lückenlosen, systemweiten Sicherheitspolitik möglich. Ideal wäre eine flexible, zentral administrierbare und sich an der Organisationsstruktur ausrichtendende Zugriffskontrollstrategie, eine solche Strategie verfolgt das RBAC Modell.

9.2 Zugriffskontrollstrategien

9.2.1 DAC

Bei Discretionary Access Control (DAC) vergibt der Eigentümer die Zugriffsrechte auf seine Objekte, er allein kann bestimmen, welcher Benutzer mit welchen Rechten auf sie zugreift. DAC ist der Standard bei den beiden am meisten verbreiteten Betriebssystemen, MS Windows und den diversen Unix Varianten. Fatales Problem bei der Zugriffskontrolle mit DAC ist, dass sich damit keine Sicherheitspolitiken durchsetzen lassen: Sie können von den Benutzern beliebig unterlaufen werden, da sie ihre Zugriffsrechte beliebig setzen können.

9.2.2 MAC

Bei Mandatory Access Control werden die Zugriffsrechte zentral vom System vergeben, die Benutzer können diese Rechte nicht mehr verändern. Um Benutzern die Rechtevergabe an ihren eigenen Objekten zu ermöglichen, ist es sinnvoll, eine Kombination aus MAC und DAC zu Benuntzen, wobei die MAC Strategie DAC überlagert. Konkret könnte ein Benutzer versuchen, eine Datei zum lesen für andere freigeben. Wenn aber die MAC Strategie ihm dies verbietet, so kann er diese Aktion nicht ausführen.

9.3 RBAC

Role-based Access Control (RBAC) bricht mit der Zuordnung von konkreten Rechten an Benutzern. Statt dessen werden die Zugriffsrechte mit Rollen verknüpft, die dann den Benutzern zugewiesen werden. RBAC wird in Ansätzen durch Benutzergruppenkonzepte wie etwa bei MS Windows realisiert, dort verfügen z.B. alle Mitglieder der Gruppe Administratoren über sehr weitreichende Zugriffsrechte, sie können auf alle Dateien beliebig zugreifen und Änderungen an der Systemkonfiguration vornehmen. Das RBAC Konzept geht jedoch weiter, mittels RBAC ist es beispielsweise möglich, auch Administratoren in ihrer Allmacht einzuschränken. Konkret könnte etwa einem Administrator untersagt werden, die EMails von anderen Benutzern einzusehen. RBAC kann mit MAC oder DAC kombiniert werden, in diesem Fall überlagert RBAC die anderen Strategien.

9.3.1 Geschichte

RBAC wurde 1992 im Rahmen der 15th National Computer Security Conference von David Ferraiolo und Rick Kuhn vorgestellt [1], beide Autoren beschäftigen sich mit diesem Thema am Computer Security Resource Center des NIST (National Institute of Standards and Technology) und halten auch einige Patente auf RBAC Implementationsideen. Im Februar 2004 wurde RBAC als offizieller Amarican National Standard ANSI INCITS 359-2004 verabschiedet.

9.3.2 Vorteile

- Zentrale Administration
  Durch die zentrale Administration aller Zugriffsrechte ergibt sich eine Zeit- und damit auch Kostenersparnis, es müssen nicht mehr Zugriffsrechte auf allen Rechneren einzeln verwaltet werden, sondern sie können einfach zentral administriert werden. So wird auch die Gefahr einer versehentlichen Fehlkonfiguration stark verringert, denn schnell können sich durch Flüchtigkeit Fehler bei der wiederholten Konfiguration von sehr vielen Rechnern einstellen, es kann auch passieren, dass einige Rechner versehentlich vergessen werden. Es ist ebenfalls einfacher, eine einzige zentrale Zugriffsrechtekonfiguration zu prüfen als viele verschiedene, auf mehrere Rechner verteilte.

- Vereinfachte Administration
  Das Rollen-Konzept vereinfacht das Hinzufügen eines Benutzers zum System sehr stark, im idealfall muss einfach nur der Benutzer mit seinen Rollen konfiguriert werden, auf die komplizierte Vergabe von zahlreichen Zugriffsrechten kann verzichtet werden (siehe Abbildung 9.1). So können ebenfalls versehentliche Konfigurationsfehler vermieden werden.

- Rollen-Orientierung
  Durch die Rollen-Orientierung, die Zugriffsrechte anhand von Rollen des Benutzers in der Organisation vergibt, ist die Rechtevergabe intuitiv, einfach und schnell einzurichten und bei späterer Prüfung logisch nachvollziehbar. Die Kompetenzen eines Benutzers und die daraus resultierende Zugriffsrechte sind durch seine Rolle in der Organisation bereits vorgegeben, daher können

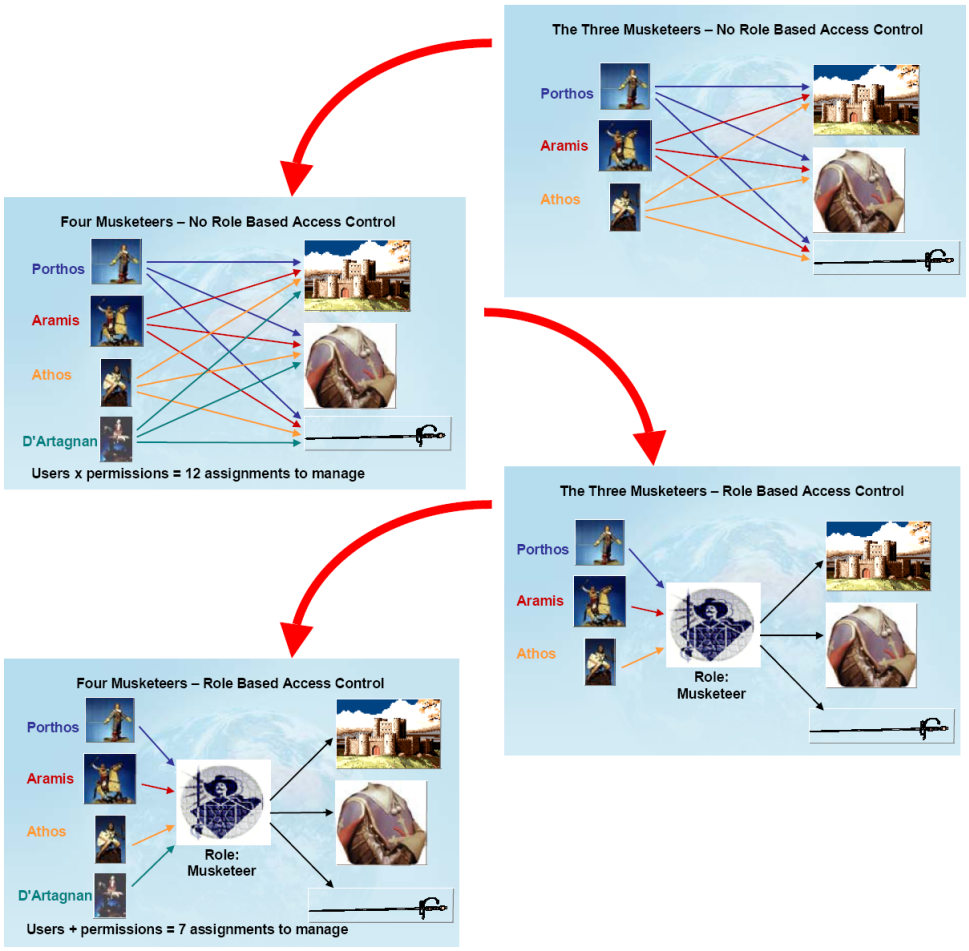


Abbildung 9.1: Beispiel - 3 Musketiere

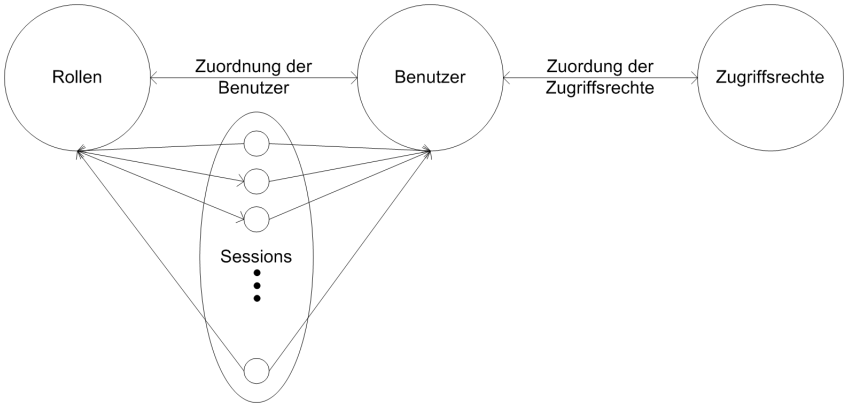


Abbildung 9.2: $RBAC_0$

die Rechte für die verschiedenen Rollen sehr einfach und sicher konfiguriert werden. Insbesondere das Grundprinzip der minimalen Rechtevergabe wird durch das Rollenkonzept sehr vereinfacht, die Zugriffsrechte für die verschiedenen Rollen lassen sich sehr einfach nachvollziehen, Konfigurationsfehler können daher relativ leicht entdeckt und korrigiert werden.

- Policy-Neutralität
  RBAC ist Policy-Neutral, es eignet sich zur Umsetzung verschiedener Sicherheitspolitiken, die durch die konkrete RBAC-Konfiguration realisiert werden.

9.3.3 RBAC Modelle

$RBAC_0$

$RBAC_0$ ist das RBAC Basismodell (siehe Abbildung 9.2). Einem Benutzer sind verschiedene Rollen im System zugewiesen, von denen er sich ggf. eine Rolle für eine Sitzung (Session) auswählen kann. Mit diesen Rollen sind verschiedene Zugriffsrechte verbunden, die Rechte aller Rollen eines Benutzers werden zur Zugriffskontrolle einfach addiert. Ist mit einem seiner Rollen das benötigte Zugriffsrechte für eine gewünschte Aktion verbunden, kann sie ausgeführt werden, sonst muss sie vom System verweigert werden.

$RBAC_1$

$RBAC_1$ ist eine Erweiterung von $RBAC_0$ um Rollenhierarchien. Die Organisationsstruktur in Unternehmen sieht gewöhnlich Vorgesetzte und ihnen untergebene Mitarbeiter vor, dieses Konzept wird in $RBAC_1$ durch Rollenhierarchien realisiert. Ein Vorgesetzter hat untergebene Mitarbeiter in verschiedenen Rollen, durch das Hierarchiekonzept erbt er dann automatisch deren Rollen.

$RBAC_2$

$RBAC_2$ ist eine Erweiterung von $RBAC_0$ um Einschränkungen (Constraints). Mögliche Constraints sind sich gegenseitig ausschließende Rollen, exklusive Zugriffsrech-

te oder Kadinalitäts-Constraints wie eine maximale Rollenanzahl für einen Benutzer.

$RBAC_3$

$RBAC_3$ ist die Kombination von $RBAC_1$ und $RBAC_2$.

9.4 Implementierungen

9.4.1 Zugriffsschutz in verteilten Systemen mit CORBA

Zugriffsschutz in verteilten Systemen ist nicht nur bei fertigen Softwarepaketen wichtig. Immer mehr wünschen sich Entwickler fertige Bibliotheken, die sie bei der Entwicklung Ihrer Produkte ohne größeren Aufwand nachnutzen können.
Im folgenden wird wird kurz umrahmt, wie man RBAC bei der Implementierung mit der CORBA-Middleware einsetzen kann. Die Common Object Request Broker Architecture (CORBA) wurde von der Object Management Group (OMG) spezifiziert. Das Ziel der OMG besteht darin eine offene, herstellerunabhängige Architektur und Infrastruktur, mit der man über Netzwerke miteinander interagieren kann, bereitzustellen.
Hierbei basiert das Standard CORBA-Zugriffsmodell auf Access Control Lists. Die Rechte werden also standardmäßig mit Gruppen- bzw. normalen Benutzeraccounts verwaltet.
Um eine rollenbasierte Rechteverwaltung nutzen zu können, existiert das Projekt RACCOON. Ziel von RACOON ist ein verwaltbarer, feingranulierter Zugriffsschutz für CORBA. RACCOON kümmert sich hierbei um Handhabung von Objekten, Rechten und Benutzern in verteilten Systemen. Für die Implementierung von RBAC unter Mithilfe von RACCOON existiert die View Policy Language (VPL).

Die View Policy Language

Die VPL wurde 1999/2000 präsentiert, ist objektorientiert und rollenbasiert:

```
roles
  Mitglied;
  Redakteur;

view Lesen controls Dokument restricted_to Mitglied {
  allow
    lesen;
    suchen;
  deny
    strong loeschen;
    schreiben;
};

view Redaktionieren : Lesen restricted_to Redakteur {
```

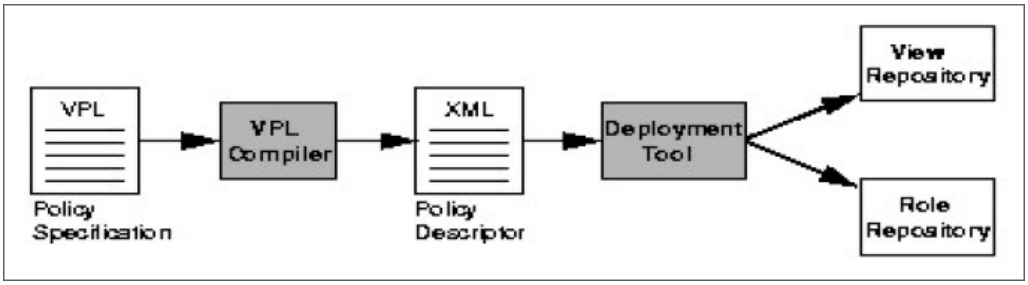


Abbildung 9.3: Integration von VPL

```
    allow
      strong schreiben;
      freigeben;
      loeschen;
};
```

In der VPL werden die möglichen Rollen mit dem Schlüsselwort roles und die möglichen Rechte mit dem Schlüsselwort view eingeleitet. In den view-Blöcken wird definiert was erlaubt und was nicht erlaubt sein soll. Obiges kurzes Beispiel definiert die beiden Rollen `Mitglied` und `Redakteur` und deren Rechte. Die Integration von VPL ist in Abbilderung 9.3 veranschaulicht. Daraus lässt sich eine allgemeine vorgehensweise ableiten

1. Erstellen einer VPL-Definition (durch Programmierer)
2. Übersetzen der Definition mit Hilfe eines VPL-Compilers (durch Programmierer)
3. Syntaxprüfung der Policy- Beschreibung & Erzeugung einer XML-Beschreibungs-Datei (durch Compiler)
4. Einbindung Rollen- und Viewkonstrukte in Repositories & Auflösung Namenskonflikte, Vermeidung von Redundanz (durch Deployment Tool)

Die RACCOON-Architektur

In Abbildung 9.4 ist die RACCOON-Architektur kurz veranschaulicht. Bei dem "Credential Objects„ ("Referenz, Zeugnis„-Objekte) handelt es sich um repräsentative Sicherheitsattribute des Aufrufenden Clients. Diese werden clientseitig erzeugt und gehalten und innerhalb einer Session an den Server übermittelt. Die Rolleninformationen werden über Zertifikate (X.509 v3) ausgetauscht. Die Autorisierung erfolgt hierbei über eine SSL-Verbindung und wird anhand des Datums und des Zertifikates überprüft. Die Administration erfolgt clientseitig unter Mithilfe eines Manager-GUIs.

9.4.2 Solaris

Auf UNIX-Systemen ust root der Administrations-User. Für Ihn gibt es keine Begrenzung in dem was er tun kann. Dieses Konzept stamt aus den späten 60er

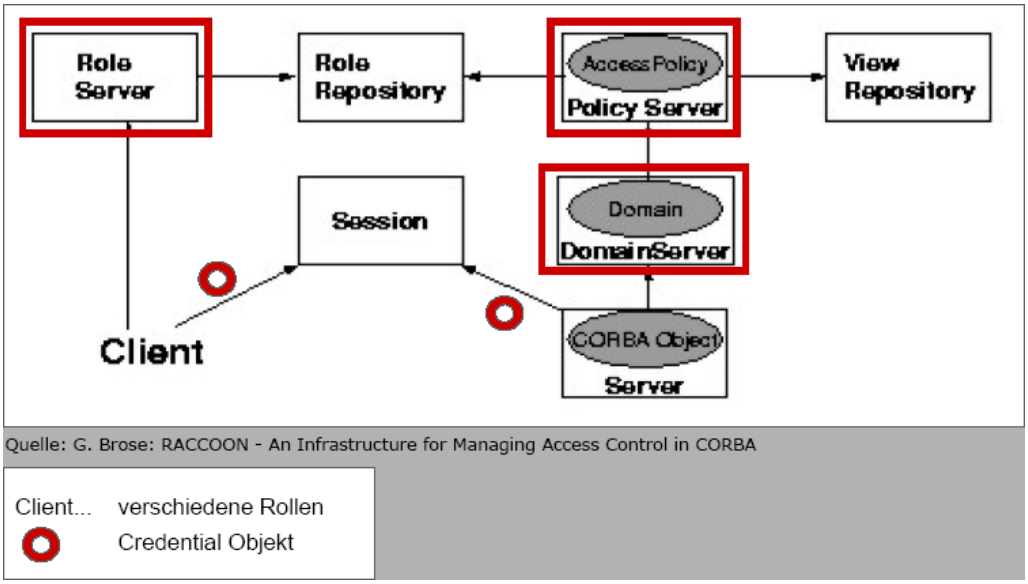


Abbildung 9.4: RACCOON- Architektur

Jahren. Heute verwendet noch nicht einmal Microsoft ein solch undifferenziertes Administrationskonzept. Selbst hier lassen sich Gruppen mit Sätzen von Privilegien einrichten, welche sich durch Zugehörigkeit einzelner User zu den Gruppen steuern lassen.

RBAC bei Standard-Solaris

Bisher war es in UNIX-Systemen immer so, dass ein Benutzer in der /etc/passwd angelegt wurde und eine eindeutige User-ID (uid) erhalten hat. Zudem war dieser Benutzer meist Mitglied einer Default-Gruppe und möglicherweise mehrerer alternativer Gruppen. Die UserID und GroupID sind im Kontext der Shell hinterlegt mit der der Benutzer gerade arbeitet. Diese werden (vereinfacht ausgedrückt) mit den zu Dateien oder Verzeichnissen hinterlegten abgeglichen. Der Benutzer darf ein Programm ausführen, je nachdem wie die Zugriffsrechte (File-Permissions) auf die Datei durch den Eigentümer (oder root) eingerichtet wurden,

- wenn er entweder der Eigentümer ist,
- wenn er Mitglied der benannten Gruppe ist
- oder wenn er irgend jemand ist und jeder Zugriff besitzt.

Schwachpunkte des Konzepts In lokalen Dateisystemen sind die Zugriffsrechte für lokale Benutzer gültig[1].
Der Administrations-User (root) auf dem lokalen System, darf bei Verzeichnissen und Dateien die via NFS gemountet wurden, nur so viel wie der Administrations-User auf dem exportierenden Server definiert hat.

[1]Ein durch NIS oder NIS+ authentifizierter Benutzer (lokal angemeldet) ist für Unix ein lokaler Benutzer.

Die Benutzer sollten auf dem Client als auch Server die gleichen Namen und UserIDs besitzen.

Die Lösung durch RBAC Mit "Role based access control„ (RBAC) existiert ab Solaris 8 eine mögliche Lösung für diese Probleme[2]. Jedem Benutzer wird hierbei eine Rolle zugewiesen, welche einen Satz von mehreren Privilegien (authorizations), der die Möglichkeit es Users zum Aufruf administrativer Programme einschränkt oder erweitert, enthält.
Die Zugriffsrechte können nun so gesetzt werden, dass administrative Programme grundsätzlich von jedem gelesen und ausgeführt werden können. Der Systemkern ermittelt, durch die Prüfung der Privilegien, ob ein Benutzer ein Programm aufrufen darf (falls ja mit welchen Optionen) oder nicht. Die Zugriffsrechte bei Konfigurationsdateien können hierbei noch restriktiver gehandhabt werden.
Es ist möglich RBAC lokal, aber auch via NIS+ netzwerkweit zu konfigurieren.
SUN liefert zum einen ein Application Programming Interface (API) für Entwickler[3] und zum anderen ein Kommandozeilen-Tool für Anwender[4] mit, dass es erlaubt RBAC auch in Shell-Skripten zu verwenden.

Trusted Solaris

Trusted Solaris ist sicherheits-optimierte Version des Betriebssystems Solaris, welches von SUN entwickelt wird. Um dies umzusetzen besitzt Trusted Solaris u.a. folgende Erweiterungen:

- Abschaffung des Superusers
- erweiterte Kontrolle über Dateizugriffe
- Monitoring von Aktionen
- Abfangen von Tastatureingaben verhindern
- ...

Ein Programm benötigt um auf ein Objekt zugreifen und es bearbeiten zu können einen speziellen Schlüssel. Das Subjekt erhält eine Referenz auf ein Objekt, welches alle Rechte die das Subjekt an dem Objekt besitzt, enthält. Für diese Umsetzung werden sogenannte "Capabilities„ verwendet. Bei der Nutzung der Capability (Zugriff auf das Objekt) werden die Rechte überprüft und es wird über den Zugriff entschieden. Diese Capabilities setzen Teile der DAC-Restriktionen ausser Kraft. Dieser Ansatz ist dem Superuser-Ansatz engegengesetzt. Capabilities helfen die Root-Rechte in kleinere Abschnitte zu fassen.

[2] RBAC wird von SUN mit einem kompletten Satz von Privilegien und Profilen ausgeliefert, die für die Administration eines Systems erforderlich sind.
[3] siehe Manpage rbac(5)
[4] siehe Manpage auth(1)

Mandatory Access Controls (MAC) Bei den Mandatory Access Controls (MAC) werden die Rechte in verschiedene Ebenen mittels Labels eingeteilt. Es existieren hierarchische Sicherheitsstufen wie “public„, “private„ oder “private-engeneering„. Benutzer können Labels nicht überwinden und bleiben in ihrer Ebene. Dadurch wird verhindert das Daten unbeabsichtigt in falsche Hände geraten können.

Discretionary Access Controls (DAC) Bei den Discretionary Access Controls (DAC) werden die Rechte mit normalen Dateizugriffsrechte bzw. ACLs verwaltet. Die Steuerung des Zugriffs auf Daten erfolgt in Abhängigkeit der Benutzeridentität bzw. Gruppenmitgliedschaft. Root ist hierbei, im Gegensatz beim normalen Solaris, jedoch nicht von diesen Beschränkungen ausgenommen Die DACs werden zusammen mit den MACs für die Kontrolle des Dateisystems verwendet.

Role Based Access Control (RBAC) Die Aufteilung der administrativen Teile erfolgt auf mehrere (sich ergänzende) Administratoren. Hierbei können alle administrativen Aktivitäten überwacht, protokolliert und zurückverfolgt werden. Es existieren standardmäßig die folgenden Rollen:

- Security administrator (secadmin)
- System administrator (admin)
- Operator (oper)

Die Anmeldung erfolgt durch normales einloggen mit anschließender Übernahme der administrativen Rolle(n).

Rights Profiles Es existiert eine Datenbank mit funktionsorientierten Rechte-Profilen. Diese sind in hierarchischer Form miteinander kombinierbar. Profile können hierbei abgeändert und angepasst werden. Somit ist man in der Lage eine schnelle und effektive Verteilung von Rechten und Restriktionen zu gewährleisten.

9.4.3 J2EE

Die Java 2 Platform Enterprise Edition (J2EE) ist ein Standard für verteile, komponentenbasierte Multi-Tier Geschäftsanwendungen, konkrete Implementierungen des Standards sind z.B. Sun One oder JBoss. Wichtige Bestandteile der Spezifikation sind Enterprise JavaBeans (EJB) und Java Servlets bzw. Java Server Pages (JSP), EJBs sind verteile Objekte, die über Java Remote Method Invocation (RMI) kommunizieren, Java Servlets bzw. Java Server Pages ist eine Technologie für dynamische Webseiten.
Das Sicherheitsmodell der J2EE entspricht $RBAC_0$, vorgesehen sind Benutzer, Rollen und damit verknüpfte Rechte, nicht vorgesehen sind Rollenhierarchien und Constraints.
Man kann bei der J2EE zwischen programmatischen und deklarativen Sicherheitsmechanismen unterscheiden. Programmatische Sicherheitsmechanismen sind fest im Quellcode integriert, über die Methoden ejbContext.isCallerInRole() bzw.

httpServletRequest.isUserInRole() kann innerhalb von Methoden die Rollenzugehörigkeit des gegenwärtigen Benutzers geprüft werden. Deklarative Sicherheitsmechanismen werden über Deployment Descriptors, das sind standardisierte XML Konfigurationsdateien, realisiert. In ihnen werden Benutzer und Rollen definiert und Zugriffsberechtigungen für Methoden gesetzt.

9.4.4 Siemens DirX

Siemens DirX ist ein sogenanntes Identity Managementsystem, es dient zur zentralen Autentisierung von Benutzern und zur Autorisierung von deren Zugriffsrechten, es wird für MS Windows, Sun Solaris und diverse Unix Platformen angeboten. Die Directory-Server Komponente des Systems ist LDAP/X.500 konform (X.500 Metadirectory, LDAP als Abfrageprotokoll), die Komponente DirXmetaRole ermöglicht die RBAC-konforme Administration von Zugriffsrechten.
Das DirX System passt dazu die vom Administrator spezifizierten Zugriffsrechte an Betriebssystem- oder Anwendungsspezifische Rechte an, die dann über ein Meta-Directory an die Endsysteme weitergegeben werden. Die eigentliche Zugriffskontrolle bleibe dabei auf dem Endsystem, DirX kümmert sich nur um die Verwaltung und Verteilung von Rechten, nicht aber um deren konkrete Durchsetzung.

9.5 Ausblick

RBAC ist ein möglicher Ansatz zur Vereinfachung der Sicherheitsadministration, der vor allem von NIST gefördert wird. Prinzipiell erscheint das RBAC Konzept von Rollen mit daran gebundenen Rechten sinnvoll, ob es sich in der Praxis durchsetzt hängt von der Qualität der verfügbaren Implementierungen ab.
Gegenwärtig gibt es noch kein allumfassendes RBAC System, welches Zugriffsrechte sowohl auf Betriebssystemebene (Dateizugriff, Netzwerkzugriff) als auch auf Anwendungsebene durchsetzt, es gibt keine standardisiertern Rollen- und Rechtedeklarationsschemata und keine einheitlichen Schnittstellen für die die Realisierung eines RBAC Zugriffskontrollsystems unabhängig von Betriebssystem, Middleware und Programmiersprache.
Dennoch wird das RBAC Konzept etwa in der J2EE Middleware erfolgreich umgesetzt, es kann eine möglich Lösung for das Problem der komplizierten Sicherheitsadministration sein, ob es Die Lösung der Zukunft ist wird sich zeigen.

Literaturverzeichnis

[1] David F. Ferraiolo, D. Richard Kuhn. Role-Based Access Control. *15th National Computer Security Conference*, 1992.

[2] David F. Ferraiolo, D. Richard Kuhn & Ramaswamy Chandramouli. *Role-Bases Access Control*. Artech House Inc., 2003. ISBN 1-58053-370-1.

[3] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode, Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock. SUN J2EE Tutorial. *SUN*, 2004. http://java.sun.com/j2ee/1.4/download.html#tutorial.

[4] Manuel Dräger. Simulation einer Bankverwaltung mit der Middleware CORBA. *Belegarbeit bei Prof. Dr. Horn*, 2004. http://burns.cs.uni-potsdam.de/~mdraeger/gse3.pdf.

[5] N/A. Siemens Dirx Solutions - Totally Integrated Identy Management. *Siemens White Paper*, 2003. http://www.siemens.com/Daten/siecom/HQ/ICN/Internet/Meta_Directory/WORKAREA/meta_dir/templatedata/English/file/binary/DirX_Solutions_2003_1072421.pdf.

[6] N/A. Siemens Meta Directory. *Siemens White Paper*, 2003. http://www.siemens.com/Daten/siecom/HQ/ICN/Internet/Meta_Directory/WORKAREA/meta_dir/templatedata/English/file/binary/MD_WhitePap_April03_1072418.pdf.

[7] Nicolas Noffke. Entwurf und Implementierung eines Rollendienstes für die Zugriffskontrolle in CORBA. *Diplomarbeit bei Prof. Dr. Löhr*, 2001.

[8] Tobias Grave, Nils Illensee, Marco Puhlmann. EJB Security. *Belegarbeit bei Prof. Dr. Horn*, 2003. Sicherheitsaspekte bei der Konstruktion von Enterprise Java Beans Applikationen.