Management von Sicherheitsdiensten in verteilten Systemen

*von Michael Steiner, Günter Karjoth, Ralf Hauser**

Zusammenfassung: *Immer mehr wird in vernetzte Arbeitsplatzrechner investiert, um außer Benutzerfreundlichkeit auch den Nutzen billiger Hardware und Standardsoftware zu haben. Neben Sicherheit ist deren Management eine der wichtigsten Anforderungen, welche in solch verteilten Systemen gestellt werden. Insbesondere sind Werkzeuge gefordert, die große Benutzerpopulationen und von ihnen zugreifbare Betriebsmittel uniform verwalten. Wir stellen die Sicherheitsmanagement -Architektur des RACE II Projektes SAMSON vor. Besonderes Augenmerk liegt auf der Sicherung der Infrastruktur und der Verwaltung heterogener Verzeichnisse, welche die autorisierten Benutzer und Betriebsmittel definieren.*

1 Einführung

Verteilte Systeme, die auf Netzen von nicht nur lokalem, sondern zunehmend auch globalem Einzugsbereich basieren, werden immer alltäglicher und scheinen die Rechnersysteme der Zukunft zu sein. Benutzer verteilter Systeme können von ihrem Arbeitsplatzrechner aus auf Betriebsmittel zugreifen, welche über die ganze Welt verteilt sind. In solchen komplexen und heterogenen Umgebungen können zahlreiche Mißbräuche die Integrität und Vertraulichkeit der Daten bzw. die Privatsphäre des Benutzers in Frage stellen [2]. Um derartigen Mißbräuchen zu begegnen, existieren bereits verschiedene Sicherheitsverfahren.

Authentifikation spielt dabei eine wesentliche Rolle. In traditionellen Großrechnern basiert die Authentifikation auf lokal verfügbarer Information über die Benutzer und ihre Passwörter. Mit steigender Verteilung der Rechner nimmt die Authentifizierung jedoch einen globalen Charakter an, da man nicht mehr länger davon ausgehen kann, daß diese Information lokal auf allen Knoten des Systems zur Verfügung steht bzw. sicher verwaltet werden kann. Systeme wie Kerberos [3] oder KryptoKnight [4] verwirklichen verteilte Authentifikation bereits heute.

Da diese Dienste nebeneinander bestehen können, muß der Systemverwalter Werkzeuge zur Hand haben, die es ihm erlauben, sie über alle ihm unterstehenden Rechner hinweg in einheitlicher Form zu verwalten und zudem den gleichen Sicherheitsrichtlinien zu unterwerfen. Zum Beispiel sollte der Benutzer 'Hans Meier' als eine Person wahrgenommen und verwaltet werden, auch wenn er verschiedenen Authentifikationssystemen unterstellt und in verschiedenen Datenformaten in den jeweiligen Benutzerverzeichnissen zu finden ist.

Das Projekt SAMSON, "Security and Management Services in Open Networks", entwickelt einen Ansatz zur einheitlichen Verwaltung von Sicherheitdiensten. In diesem Artikel stellen wir die SAMSON-Architektur vor und konzentrieren uns dabei auf die Verwaltung von heterogenen Verzeichnissen, welche die Benutzer und deren Zugriffsrechte auf Betriebsmittel definieren. Zuerst werden in Kapitel 2 die Ansprüche an die Sicherheitsverwaltungsarchitektur vorgestellt. Kapitel 3 befaßt sich dann kurz mit bereits bestehenden Ansätzen. Die SAMSON-Architektur wird in Kapitel 4 vorgestellt. Der Frage, wie auch die Management-Infrastruktur gesichert werden kann, gehen wir in Kapitel 5 nach. Das Management von Authentifikation wird in Kapitel 6 diskutiert. Kapitel 7 faßt die Ergebnisse zusammen und schließt den Artikel mit einem Ausblick auf zukünftige Arbeiten ab.

Der Anhang dieses Artikels enthält technische Daten und Hinweise auf weitere Veröffentlichungen zum SAMSON Projekt.

2 Zielsetzungen

Bestehende Datensammlungen beinhalten Zugriffsinformation, wie z.B. Benutzernamen, Passwörter und Zugriffskontrollisten, auf die Sicherheitsfunktionen aufgesetzt sind. Es ist die Aufgabe des *Sicherheitsadministrators*, diese Informationen überall im System mit Hilfe von anwendungsspezifischen Verwaltungsdiensten zu installieren.

Mit dem Trend zu immer größeren und heterogenen Systemen werden die Vielzahl und Besonderheiten all dieser „ad-hoc" Entwürfe zu einer Plage für den Systemadministrator. Er ist gezwungen, sich mit einer steigenden Zahl von Unzulänglichkeiten auseinander zu setzen: Werkzeuge mit vergleichbarer Funktion werden unabhängig voneinander entworfen und arbeiten auf verschiedenen Informationsbasen und mit inkompatiblen Benutzerschnittstellen. Da zudem in vielen Fällen nur lokaler Zugriff auf Daten möglich ist, wird eine zentrale Verwaltung ausgeschlossen.

Nebst großem Aufwand stellt dies ein hohes Potential an Inkonsistenzen und somit ein erhöhtes Risiko von Sicherheitslücken dar. Es ist deshalb notwendig, eine konsi-

*IBM Forschungslabor Zürich, Säumerstr. 4, CH-8803 Rüschlikon, Schweiz

Dieser Artikel ist eine überarbeitete Fassung unseres Beitrages für die Konferenz „Sicherheit in Informationssystemen" [1].

stente, integrierte Lösung zu entwickeln, welche folgende Anforderungen erfüllt:

- *Einheitliche Schnittstelle*: Alle heterogenen Systemmanagement-Funktionen werden basierend auf Normen für offene Netze und Verwaltungstechnologie in Einklang gebracht.
- *Erweiterbarkeit*: Zusätzliche Funktionen, wie z.B. logging, audit, recovery, müssen leicht integriert werden können.
- *Modularität*: „Steckerkompatible" Komponenten werden durch generische Schnittstellen (API) mit dem Anwendungsprogramm verbunden.
- *Skalierbarkeit*: Eine hierarchische Organisation vermeidet zentralisierte Komponenten, deren Kapazität überschritten werden könnte.
- *Sicherheit*: Die Integrität und die Vertraulichkeit der Managementfunktionen wird geschützt.
- *Sicherheitsrichtlinien:* Die Durchsetzung der Sicherheitsrichtlinien wird durch Grundwerte (defaults) und Parametrisierung erzwungen.
- *Verteilung der Pflichten:* Die mißbräuchliche Anhäufung von Kompetenzen wird durch eine klare Aufteilung der Rollen verhindert.

Vorschläge für Sicherheitsmanagement-Architekturen sollten an diesen Kriterien gemessen werden.

3 Managementmodelle

Sicherheitsmanagement in heterogenen, verteilten Systemen ist ein neues Feld, welches in vielen verwandten Gebieten verwurzelt ist. Ziel ist es, Vorteil aus der Verfügbarkeit bestehender Systeme, Management-Werkzeuge und Protokolle zu ziehen, indem sie auf die spezifischen Probleme von Sicherheitsmanagement angewendet werden.

In [5] definiert ISO einen Rahmen für sicherheitsrelevante Normen, in dem Begriffe und Dienste vorgegeben sind. Die OSI-Sicherheitsarchitektur besteht aus drei Schichten von Managementfunktionen: Systemsicherheit, Sicherheitsdienste und Sicherheitsmechanismen.

Systemsicherheit umfaßt die Verwaltung von Sicherheitsrichtlinien und die Interaktion mit anderen OSI-Managementsystemen. Sicherheitsdienste schließen Authentifikation, Zugriffskontrolle, Vertraulichkeit, Datenintegrität und Beglaubigung ein. Sicherheitsmechanismen sind die Bausteine, aus denen Sicherheitsdienste realisiert werden. Verschlüsselung, digitale Unterschrift, Zugriffskontrolle, Datenintegrität, Authentifikation und Beglaubigung sind Beispiele spezifischer Mechanismen; Trusted Functionality, Security Labels, Security Audit Trail und Sicherheits-Recovery sind allgemeine Mechanismen.

Das „OSI Authentication Framework" [6] stellt Grundlagen zur Verfügung, welche zur Implementierung von Authentifikationsdiensten und -mechanismen verwendet werden können. Es stellt aber keine Protokolle und API's zur Verfügung, welche Portabilität erlauben und Kompatibilität gewähren würden.

Um eine Modellierungsgrundlage und Kommunikations-Infrastruktur zur Verfügung zu haben, gilt es, geeignete Werkzeuge, Mechanismen und Protokolle auszuwählen. Es muß möglich sein, Managementobjekte zu definieren, welche die Sicherheitselemente der zu verwaltenden Betriebsmittel darstellen können und zugleich von ihren spezifischen Eigenarten abstrahieren. Managementoperationen müssen auf diesem abstrakten Niveau ausgeführt werden können. Dies kann durch *Objektorientierung* erreicht werden. Da zudem moderne Anwendungen und Betriebssysteme immer mehr nach objektorientierten Prinzipien gestaltet werden, erlaubt dies auch eine bessere Integration in die verwaltete Ressource und könnte in der Zukunft zur Verschmelzung von Management- und Zielanwendungen führen.

Management in OSI (CMIP) und Internet (SNMP) sind gekennzeichnet durch ihren unterschiedlichen Grad an Objektorientierung. Gegenüber der Einfachheit von SNMP betont OSI Mächtigkeit und Flexibilität.

Sowohl SNMP als auch OSI stützen sich auf das wohlbekannte Client-Server Modell ab: der Manager (Client) beauftragt über generische Protokolle den räumlich getrennten Agenten (Server), Operationen auf den Managementobjekten auszuführen. Beide Systeme erlauben es aber, daß Agenten den Manager mit *event reports* (trap) über asynchrone Ereignisse informieren.

Ein neuerer und vollkommen objektorientierter Ansatz wird von OSF verfolgt. In ihrem DME (Distributed Management Environment), welches auf CORBA (Common Object Request Broker Architecture)[7] basiert, operieren räumlich transparent Objekte auf anderen Objekten. DME stützt sich aber sehr stark auf DCE ab und befindet sich zudem noch in der Entwicklungsphase.

4 Die SAMSON-Architektur

SAMSON baut auf dem OSI-Managementmodell auf [8]. Abbildung 1 zeigt die SAMSON-Architektur. Lokal zur Ressource verwaltet ein *Agent* die abstrakten Objekte und wendet die vom *Manager* gewünschten „Methoden" auf sie an. Systemspezifischer *Sponsor-Code* bildet dabei die abstrakte Operation in Funktionsaufrufe der Management-Schnittstelle des realen Objektes ab. Agenten und Manager kommunizieren über XMP, eine von X/Open definierte Schnittstelle zu CMIP, dem OSI Kommunikationsprotokoll zum Austausch von Managementinformationen. Dies erlaubt es, Plattformen und Produkte verschiedener Hersteller portabel miteinander zu verbinden und bereits bestehende Managementfunktionen von einem entfernten Rechner aufzurufen.

Scoping und *Filtering* erlaubt eine Auswahl von Managementobjekten, so daß mit einer Aktion Operationen auf verschieden Objekten ausgeführt werden können.

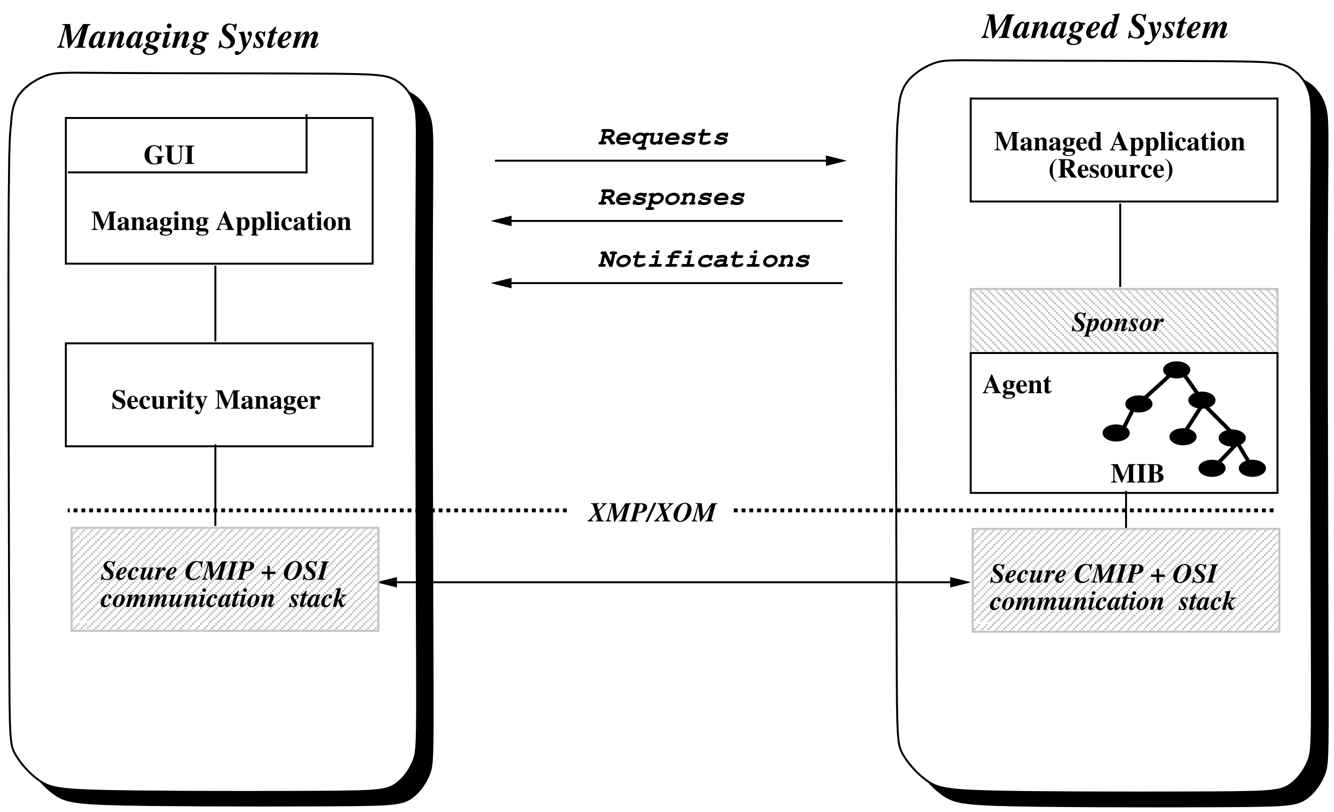

Abb. 1: Die SAMSON-Architektur

Eine große Anzahl von Agenten kann somit durch eine viel kleinere Zahl von Managern verwaltet werden. Die Manager wiederum können in einer hierarchischen Struktur organisiert werden und gegenüber der ihnen übergeordneten Managern die Rolle eines intelligenten Agenten annehmen.

Jeder Sicherheitsdienst ist als eine Menge von Managementobjekten modelliert. Sie bilden die *Management Information Base* (MIB). Zur Beschreibung dieser Objekte wird die Meta-Sprache GDMO verwendet [9]. Im Anhang ist die Spezifikation der Objektklasse *GenericManagedObject*, von der für die Authentifikation wichtige Objekte abgeleitet werden. Die Definition einer Objektklasse beinhaltet die Definition ihrer typisierten Attribute und charakterisiert die Eigenschaften wie Wertebereich und erlaubte Operationen und den Status des Objektes. Eine detaillierte Design-Spezifikation der SAMSON-Architektur in Form von GDMO-Beschreibungen befindet sich [10]. Neben der Definition der Managementinformation führt dieses Dokument die Managementanwendungen und die Objektsponsoren auf. Sie bilden die Grundlage, von der spezifischere Sicherheitsmechanismen abgeleitet werden.

Neben Vererbung ist die Enthalten-Sein-Relation ein weiterer Strukturierungsmechanismus. Jedes Managementobjekt ist in der Enthalten-Sein-Hierarchie mit der Wurzel *system* eingebunden. Dies ergibt einen Namensbaum mit global eindeutigen Namen. Abbildung 2 zeigt einen Ausschnitt aus der Enthalten-Sein-Hierarchie der in SAMSON definierten und verwalteten Objekte. Ebenso wurden für andere Dienste und Mechanismen, wie z.B. Zugriffskontrolle, abstrakte Objekte spezifiert.

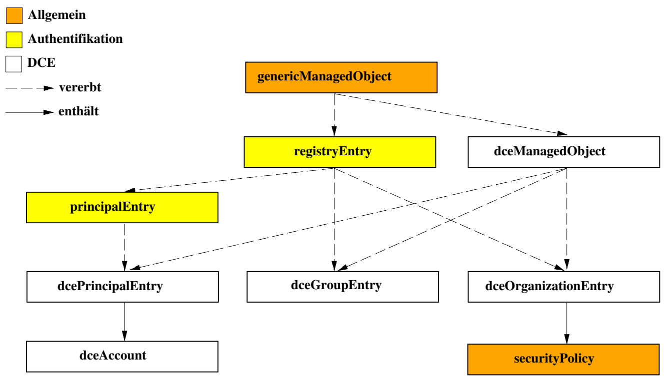

Abb. 2: Ausschnitt aus der Objekt-Hierarchie

Das Konzept des Sponsors, der die reale Ressource unter einer generischen Schnittstelle verbirgt, beläßt den Agenten und den Manager allgemein und erlaubt sowohl Änderungen als auch die jederzeitige Integration neuer Objekte. Die Zweiteilung des Agenten in einen generischen Teil, der CMIP-Anfragen vorverarbeitet und über einen Callback den Sponsor aufruft, und den Sponsor, der die eigentliche Ressource verwaltet, führt dazu, daß nach einer Portierung des Kerns nur wenige, spezifische Teile zusätzlich angepaßt werden müssen.

Auf der Implementierungsseite ist der größte Teil der

Management-Infrastruktur unabhängig von der Art der Managementobjekte. Mit Hilfe von Code-Generierungs-Werkzeugen und den GDMO-Definitionen kann eine partielle Implementierung automatisch erzeugt werden. Der einzige spezifische Teil, der für jede Objektklasse entwickelt werden muß, ist der Sponsor-Code, welcher die generischen Agentenaufrufe (call backs) in ressourcenspezifische Prozeduraufrufe übersetzt.

Policy-Regeln bestimmen den erlaubten Wertebereich und Defaultwerte von verwalteten Objekten und werden ohne Einwirken eines Administrators verbindlich angewendet [11]. Sie sind selber als Objekte modelliert und können gruppiert werden, was ein einfaches Austauschen von Regeln erlaubt. Die Objekte lassen sich in einzelne Verantwortungsbereiche aufteilen. Deren strenge Abgrenzung erzwingt die Zugriffskontrolle der Infrastruktur.

SAMSON schließt die Lücken in den OSI-Normen bezüglich der Sicherheit der Management-Infrastruktur, indem es die Managementanwendungen authentifiziert, die Berechtigung zum Zugriff auf verwaltete Objekte prüft und mittels der Audit-Funktion die relevanten Ereignisse protokolliert und die notwendigen Maßnahmen ausführt. Informationen werden integritätsgeschützt und verschlüsselt über das Netz geschickt.

Der Manager bietet dem Sicherheitsadministrator eine mächtige und benutzerfreundliche graphische Schnittstelle, die dank Scoping und Filtering einfaches Manipulieren der verwalteten Objekte erlaubt.

5 Sichern der Management-Infrastruktur

Natürlich ist die Sicherheit der verwendeten Management-Protokolle äußerst wichtig, damit das Sicherheitsmanagement seine Aufgaben richtig durchführen kann. Leider war Sicherheit nicht eine der Vorgaben, als die wichtigsten Managementprotokolle entwickelt wurden. Die Entwickler zeigten jedoch genügend Weitblick, um etwas Platz in den Protokollköpfen zu lassen, in den später definierte Sicherheitsinformation eingefügt werden kann. Ein Beispiel für diese schrittweise „Sicherung“ ist SNMP, welches vor kurzem eine wesentliche Überarbeitung erfuhr (SNMPv2) und nun auch Sicherheitskomponenten beinhaltet [12]. ISO, auf der anderen Seite, bemüht sich gegenwärtig, Zugriffskontrolle zu normieren [13].

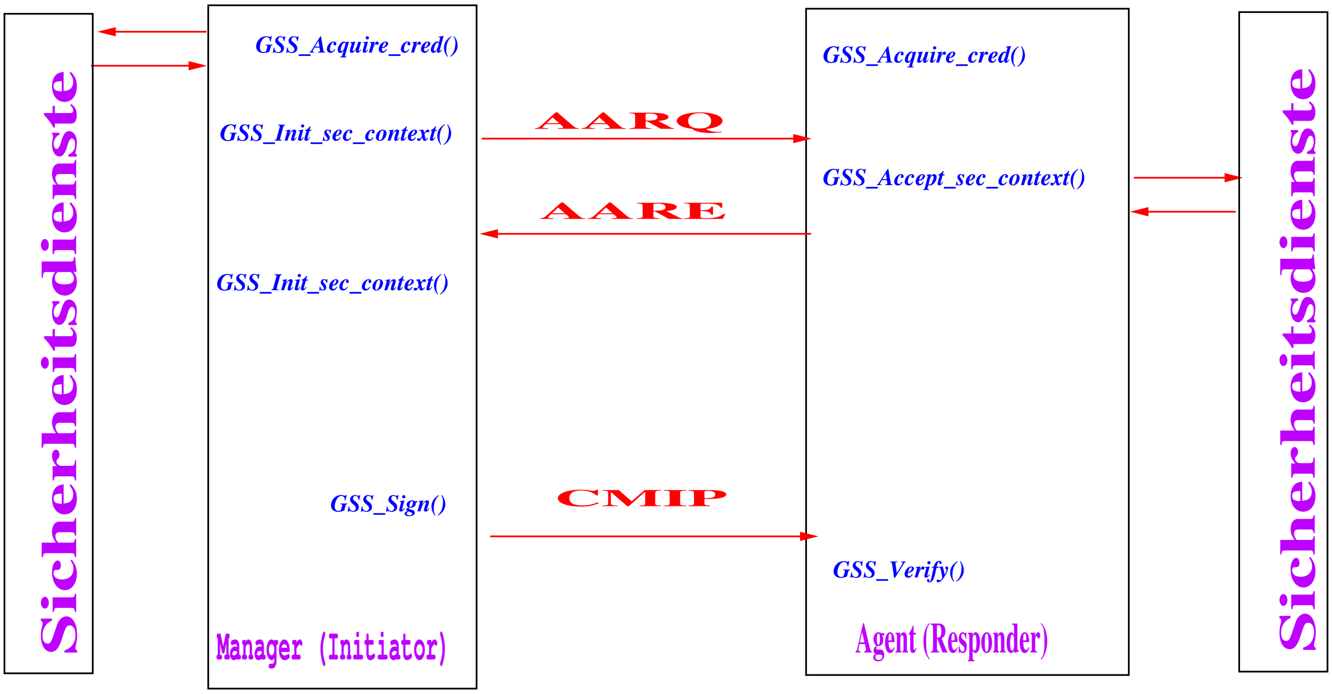

Abb. 3: Sicherheitsdienstaufrufe und Botschaftenfluß

Wenn man ein betriebsbereites System für Sicherheitsmanagement bauen möchte, wie SAMSON dies anstrebt, kann man nicht auf die Normierung unentbehrlicher Sicherheitsfunktionen warten. Deshalb wurde SAMSON durch einen Ansatz sicher gemacht, der zukünftige, normenkonforme Beiträge nicht ausschließt.

Sowohl der Manager wie auch der Agent authentifizieren sich durch lokale Aufrufe zu der GSS-API [14], welche zugrundeliegende Sicherheitsdienste wie Kerberos oder KryptoKnight verwendet. Abbildung 3 veranschaulicht die Sicherung der SAMSON-Management-Infrastruktur. Integrität oder Vertraulichkeit des CMIP-Verkehrs wird durch GSS-API *sign* oder *seal* Aufrufe realisiert.

Zugriffskontrolle ermöglicht die Objektklasse *Generic-*

ManagedObject (GDMO-Spezifikation siehe Anhang), die von jeder Klasse vererbt wird. Mit Hilfe von beglaubigten Identitätsdaten (*Privilege Access Certificates*), die in den Zugriffskontrollfeldern der CMIP PDUs eingefügt sind, und eigenen Zugriffskontrollattributen entscheidet jedes Objekt über die Authorisierung eines Zugriffes.

Letzteres steht im Gegensatz zu SNMP und zum Entwurf des OSI-Standards, welche die Zugriffskontrolle in externen Tabellen speichern. Damit benötigen jene keine Änderung der vordefinierten Objekte, erfordern aber Anpassungen der Management-Infrastruktur um die Zugriffsrechte durchzusetzen. Zudem wird auch nicht die Flexibilität von SAMSON erreicht, in verschiedenen Objektklassen unterschiedliche Mechanismen verwenden zu können.

Die Unterteilung der Sicherheitsmanagement-Funktionen in verschiedene Management-Anwendungen widerspiegelt die verschiedenen hierarchischen Managerrollen.

6 Authentifikationsmanagement

Die *Kontrolle über Angriffe*, welchen die Klienten (engl. *principals*) ausgesetzt sind, und die *Vereinbarkeit* der verschiedenen Authentifikationsmechanismen sind die Hauptanliegen für Authentifikation in heterogenen, verteilten Systemen, in denen Authentifikationsdienste von verschiedenen, voneinander unabhängigen Authentifikationsmechanismen realisiert werden. SAMSON vereinigt die Verwaltung der von ihm unterstützten Authentifikationsmechanismen in eine einzige Systemmanagementkonsole.

Um Angriffen zu begegnen, konzentriert SAMSON sich auf die Integration der Auditmechanismen in die Authentifikationsmechanismen. Selbst die neuesten Sicherheitsmechanismen stellen keine allgemeinen Auditing-Dienste zur Verfügung. Sie sind nur in der Lage, produktspezifische Fehlermeldungen zu erzeugen und können selten Gegenmaßnahmen vorschlagen. Damit die Schnittstellen der bereits eingesetzten, produktspezifischen Authentifikationsmechanismen zumindest teilweise weiterverwendet werden können, muß eine transparente Berichterstattung über Attributwertänderungen und andere Ereignisse an die Sicherheitsmanagement-Infrastruktur aus Konsistenzgründen realisiert werden.

Das Betriebsmittel muß mit der Fähigkeit, *event reports* zu erstellen, ausgestattet sein: *alle* sicherheitsrelevanten Ereignisse, z.B. erfolgreiche Logins oder fehlgeschlagene Logins, müssen an den Agenten über geeignete Callbacks gemeldet werden, um von dort durch die *event report*-Dienste an den Administrator weitergeleitet zu werden. Nicht alle Ereignisse sind jedoch von Interesse. Um zu vermeiden, daß Reports für jedes einzelne Ereignis empfangen werden, kann der Sicherheitsadministrator einen Filter (*Event Forwarding Discriminator*) konfigurieren, der in der Nähe des Betriebsmittels angesiedelt ist und nur spezifizierte Mitteilungen durchläßt. Damit wird auch keine Übertragungskapazität durch die Ausstattung des Betriebsmittels mit Auditing-Fähigkeiten unnötig verbraucht.

Nicht alle Betriebsmittel können den Sicherheitsdiensten die notwendigen Informationen zur Verfügung stellen. Der „Login Daemon" zum Beispiel mag weder eine normierte Sicherheitsschnittstelle anbieten noch die notwendigen „event reports" erzeugen. Das Fehlen der Schnittstelle läßt sich aber durch Verwendung eines systemspezifischen Login-Sponsors beheben, der regelmäßig temporäre Dateien liest, in denen relevante Auditinformationen abgespeichert werden. Wenn das Betriebsmittel den gewünschten Dienst überhaupt nicht bereitstellt, gibt es keinen anderen Weg als das Quellprogramm des Betriebsmittels mit den notwendigen Sicherheitsfunktionsaufrufen zu instrumentieren.

Eine Möglichkeit, die Konsistenz und Integration der Verzeichnisse zu erreichen, wäre der Entwurf eines Metaverzeichnisses, das jedes Attribut über jeden Benutzer und jedes Betriebsmittel kennt. Dieses könnte zentral verwaltet werden und würde an die Stelle der einzelnen Verzeichnisse treten. Dieser Ansatz mag wohl für einzelne Fälle mit ausgewählten Verzeichnissen funktionieren, läßt sich aber sicher nicht auf große Sammlungen verschiedenartigster Verzeichnisse verallgemeinern. Besser ist es, ein systemweites, erweiterbares und objektorientiertes Datenmodell zu entwickeln, in dem alle einem Benutzer oder Betriebsmittel zugeordneten Verzeichniseinträge wie Komponenten eines einzelnen *verteilten Managementobjektes* behandelt werden. Im Gegensatz zu den meisten objektorientierten Datenbanken, erzwingt dieser Ansatz keine strengen, transaktionsorientierten Konsistenzbedingungen für das zu verwaltende Objekt. Er folgt statt dessen einem lockeren „lazy-update"-Ansatz, der typisch für X.500 und andere verteilte Verzeichnisse ist.

```
authenticationMechanismPackage PACKAGE
 BEHAVIOUR
  "SAMSON Architecture":securityMechanismBehaviour,
  authenticationMechanismPackageBehaviour
  BEHAVIOUR DEFINED AS
   "this package contains attributes and notifications
   for a generic authentication mechanism.
   The attributes are derived from
   [ISO/IEC 10181-2 : 1991] chapter 7";;
 ATTRIBUTES
  cryptoTechFlag          GET-REPLACE,
  mutualAuthFlag          GET-REPLACE,
  entityAuthFlag          GET-REPLACE,
  dataOriginAuthFlag      GET-REPLACE,
  vulnerabilityClass      GET-REPLACE,
  configurationClass      GET-REPLACE,
  policyFile              GET-REPLACE,
  masterKey               GET-REPLACE,
  dateOfLastKeyChange     GET;
 NOTIFICATIONS
  "ISO/IEC 10165-2 : 1992":attributeValueChange,
  "ISO/IEC 10165-2 : 1992":stateChange;
REGISTERED AS {samson aut(3) package(4) general(1) 1};
```

Abb. 4: GDMO-Spezifikation des *Authentikationsmecha-*

nismus

Das in Abbildung 4 gezeigte *authenticationMechanism-Package* ist ein Teil des (in diesem Artikel nicht weiter beschriebenen) Managementobjektes *securityMechanism*. Das Paket charakterisiert die für Authentifikation wichtigen Mechanismen gemäß [6], definiert die Namen der relevanten Subpolicies und stellt die für die Verwaltung des Master-keys des Systems notwendigen Einrichtungen bereit.

7 Zusammenfassung und Ausblick

In diesem Beitrag haben wir einen Ansatz, wie er im SAMSON Projekt verfolgt wird, skizziert, um systematisch die sehr komplexe Thematik des Sicherheitsmanagements anzugehen. Ressourcen und Sicherheitsdienste werden als abstrakte Objekte behandelt. Sie werden in GDMO spezifiziert und sind damit für Manager und Agenten direkt zugänglich. Dies erlaubt ferner eine halbautomatische Implementierung. Erweiterungen zu neuen Objektklassen sind leicht möglich. Damit bietet SAMSON dem Administrator eine abstrakte Darstellung verschiedener Verzeichnisse an, die eine einheitliche Verwaltung erlaubt.

SAMSON zeichnet sich durch die folgenden Merkmale aus. Es basiert auf einer offenen Sicherheitsmanagement-Architektur und verwendet normierte Netzmanagement-Protokolle. Es ist fähig, verschiedene Verzeichnisse (DCE, X.500, KryptoKnight und RACF) zu verwalten. Es ist erweiterbar, um andere Sicherheitsdienste wie Zugriffskontrolle, Audit und Recovery zu verwalten. Es erlaubt ferner die Definition von Sicherheitsrichtlinien als Objekte und deren Erzwingung durch die Interaktionen zwischen Managementobjekten. Nicht zuletzt ist auch das SAMSON-Sicherheitsmanagement selbst geschützt durch Authentifizierung des Benutzers, Integrität der zu übermittelnden Daten und Zugriffskontrolle auf die Objekte in der MIB.

Um die Machbarkeit des gewählten Ansatzes zu demonstrieren, wurde im Rahmen des Projektes auch ein Prototyp entwickelt. Dieses Tool, innovativ in seiner Möglichkeit heterogene Netze zu verwalten, wurde kürzlich an der IWACA'94 in Heidelberg mit Erfolg der Öffentlichkeit vorgestellt. Zudem fließen die Erkenntnisse, die in SAMSON gemacht werden, in die Normierungsgremien von IEEE und ETSI ein.

Eine Weiterführung dieses Ansatzes würde drei Erweiterungen erfordern, die gegenwärtig nicht von SAMSON verfolgt werden: das Konzept von Ortsunabhängigkeit (CMIP erfordert die Addresse des Agenten der ein Objekt verwaltet), die Fähigkeit, Managementoperationen für „lazy-updates“ und verzögerter Bestätigung zwischenzuspeichern und die Modellierung der Regeln zur Manipulation der Objekte und ihrer Darstellung als Managementobjekte, um generische Manager und dynamisches Einbinden neuer Objekt-Typen zu ermöglichen.

Danksagung Dieser Artikel wurde zum Teil vom RACE II SAMSON Projekt, R2058, finanziell unterstützt. Stefano Zatti war aktiv an der Gestaltung von SAMSON beteiligt, und seine Ideen haben auch diesen Artikel mitgeprägt. Michael Waidner gebührt Dank für seine wertvollen Kommentare.

Literatur

[1] R. Hauser, G. Karjoth und M. Steiner. Management von Sicherheitsdiensten in verteilten Systemen. In K. Bauknecht and S. Teufel, Hrsg., *Sicherheit in Informationssystemen*, S. 7–21. Verlag der Fachvereine Zürich, 1994.

[2] M. Jurecic, U. Kohl und E. Pelikan. Datenschutz und Datensicherheit für verteilte Klinikanwendungen. In *Arbeitstreffen Entwicklung und Management verteilter Anwendungssysteme*, 14. und 15. Oktober 1993. GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme.

[3] J.G. Steiner, C. Neumann und J.I. Schiller. Kerberos: An authentication service for open network systems. In *Usenix Winter Conference*, S. 191–202, January 1988.

[4] R. Molva, G. Tsudik, E. van Herreweghen und S. Zatti. *KryptoKnight* authentication and key distribution system. In *1992 European Symposium on Research in Computer Security*, S. 155–174, 1992.

[5] OSI Reference Model – Part 2: Security Architecture. IS 7498-2, ISO/IEC JTC/SC21.

[6] Authentication framework. IS 10181-2, ISO/IEC JTC1/SC21.

[7] The common object request broker: Architecture and specification. OMG Document Number 91.12.1, Draft 10 December 1991. Revision 1.1.

[8] OSI structure of management information. IS 10165, ISO/IEC JTC1/SC21.

[9] OSI structure of management information: Guidelines for the definition of managed objects. IS 10165-4, ISO/IEC JTC1/SC21.

[10] Detailed design specifications. Deliverable D3, RACE 2058 SAMSON, April 1993.

[11] R. Hauser und S. Zatti. Security, authentication, and policy management in open distributed systems. In *10th International Information Security Conference IFIP SEC'94*, Curacao, Dutch Antilles, May 23-27 1994. Elsevier.

[12] J. Galvin und K. McCloghrie. Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2). RFC 1446, Network Working Group, April 1993.

[13] Information technology – open systems interconnection – systems management: Objects and attributes for access control. DIS 10164-9, ISO/IEC JTC 1, 1993. CCITT Rec. X.741.

[14] J. Linn. Generic security service application program interface. RFC 1508, Network Working Group, September 1993.

[15] C. Capellaro. SAMSON – managing distributed assets. In *Datapro Conference*, Nizza, 1993.

[16] S. Lechner. SAMSON: Management of security in open systems. *Computer Communications*, 17(7):538–543, July 1994.

[17] B. Baum-Waidner, H. Bunz und C. Capellaro. SAMSON, security management in a health care scenario. In H.-J. Kugler, A. Mullery und N. Niebert, Hrsg, *Towards a Pan-European Telecommunication Service Infrastructure – IS&N'94*, Lecture Notes in Computer Science #851, Seiten 503–512. Springer-Verlag, 1994.

Informationen über das RACE II SAMSON Projekt

Das SAMSON-Projekt wird von der Kommission der Europäischen Gemeinschaft für drei Jahre (1992–94) gefördert. Das Konsortium besteht aus den folgenden Mitgliedern: Siemens AG (Konsortiumsführer), Bull, CSELT, GMD, IBM, ICL, SIRTI und TELIA.

In den folgenden Deliverables beschreibt SAMSON umfassend die erarbeiteten Ergebnisse:

D1 Functional Requirements

D2 Top Level Specification

D3 Detailed Design Specification

Kurzfassungen und „executive summaries" der Deliverables sind über „anonymous ftp"

`samson.darmstadt.gmd.de:/pub/samson`

erhältlich. Weitere Publikationen über SAMSON sind [11], [15], [16] und [17].

Anhang: GDMO-Spezifikation des *GenericManagedObject*

Diese Spezifikation definiert die Klasse *genericManagedObject* mit ihren drei Attributen. Sie ist eine direkte Unterklasse von *TOP*, definiert in der Norm ISO/IEC 10165-2, und Wurzel aller SAMSON-Objektklassen. Eigenschaften der Klasse und ihrer Attribute sind in den jeweiligen *Behaviour*-Komponenten beschrieben.

```
genericManagedObject MANAGED OBJECT CLASS
  DERIVED FROM "ISO/IEC 10165-2 : 1992":top;
  CHARACTERIZED BY genericManagedObjectPackage PACKAGE
  BEHAVIOUR genericManagedObjectBehaviour BEHAVIOUR DEFINED AS
    "a generic managed object";;
  ATTRIBUTES
    aCLAdministratorList GET-REPLACE ADD-REMOVE,
    relevantPolicyRuleList GET-REPLACE ADD-REMOVE,
    objectName GET;
  NOTIFICATIONS
    "ISO/IEC 10165-2 : 1992":objectCreation,
    "ISO/IEC 10165-2 : 1992":objectDeletion,
    "ISO/IEC 10165-2 : 1992":attributeValueChange,
    "SAMSON Architecture":serviceReport;;;
REGISTEREDAS{ samson acl(2) managedObjectClass(3) general(1) 1};

aCLAdministratorList ATTRIBUTE
  WITH ATTRIBUTE SYNTAX AttributeModule.ACLAdministratorList;
  MATCHES FOR SET-COMPARISON, EQUALITY;
  BEHAVIOUR aCLAdministratorListBehaviour BEHAVIOUR DEFINED AS
    "This attribute provides the list of individuals (e.g. the system
    super-user, the owner of the object or those who are granted the
    change permission) who can perform the modification of the ACI. The first
    of the list identifies the manager (the GSM) that can modify the values of the attribute itself.";;
REGISTERED AS{ samson acl(2) attribute(7) general(1) 1};

relevantPolicyRuleList ATTRIBUTE
  WITH ATTRIBUTE SYNTAX AttributeModule.RelvtPolRulListType;
  MATCHES FOR SET-COMPARISON, SET-INTERSECTION, EQUALITY;
REGISTERED AS{ samson acl(2) attribute(7) general(1) 2};
```

```
objectName ATTRIBUTE
   WITH ATTRIBUTE SYNTAX AttributeModule.ObjectName;
   MATCHES FOR EQUALITY;
   BEHAVIOUR objectNameBehaviour BEHAVIOUR DEFINED AS
      "This attribute provides the identifier of the object to be protected by the DACL.";;
REGISTERED AS{ samson acl(2) attribute(7) general(1) 3};
```

Michael Steiner studierte Informatik an der ETH Zürich. Seit 1993 ist er Doktorand an der Universität Twente (NL) und arbeitet gleichzeitig im IBM Forschungslabor Zürich. Nebst der Arbeit am RACE II Samson Projekt beschäftigt er sich mit elektronischem Zahlungsverkehr und dessen Einbindung in weltweite Informationssysteme.

Dr. Günter Karjoth studierte Informatik an der Universität Stuttgart, wo er auch promovierte. Seit 1986 ist er als wissenschaftlicher Mitarbeiter am IBM Forschunglaboratorium Zürich tätig. Seine Forschungsinteressen sind Sicherheit in verteilten Systemen und die Anwendung formaler Methoden zur Spezifikation und computerunterstützten Verifikation nebenläufiger Systeme.

Ralf Hauser studierte Wirtschaftsinformatik an der Universität Zürich. 1991 erwarb er an der Universität Toronto den Master of Science und arbeitet seither am RACE II Samson Projekt für das IBM Forschungslabor Zürich. Ausserdem ist er Doktorand und Assistent am Institut für Informatik der Universität Zürich. Seine Forschungsinteressen umfassen Softwarelizenzierung, -verteilung und -integritätsschutz, verteilte Informationssysteme (WWW) und deren Sicherheitsaspekte.