JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Visualisierung von Rechtestrukturen in Windows 2000 Netzwerken

DIPLOMARBEIT

zur Erlangung des akademischen Grades

DIPLOMINGENIEUR

in der Studienrichtung

INFORMATIK

Angefertigt am *Institut für Informationsverarbeitung und Mikroprozessortechnik*

Betreuung:

o. Prof. Dr. Mühlbacher Jörg R.

Dipl.-Ing. Hörmanseder Rudolf

Eingereicht von:

Zarda Gerald

Linz, *April 2002*

Abstract

The need for security in information technology rises steadily. In this case, there are a lot of different aspects of security to consider. One of them is in fact the internal security, which is erroneously deemed to be of secondary importance. However, what is the logic of protecting a companies´ local area network against attacks from outside when the real threat comes from inside?

This is where the *SAT*-project addresses the point. The primary objective of the *SAT* system is to reduce the complexity of access right control and thus offer support for administrators to ensure the internal security. Unfortunately standard tools of the Windows 2000 operating system do not provide an efficient way for administrators to control access rights and permissions. Specifically for this reason the *Security Analysis Tool 2 (SAT 2)* has been developed.

Within the scope of this diploma thesis, a *Microsoft Management Console (MMC) Snap-in* prototype for visualizing the rights structures in a network has been implemented. The *SAT2* system allows a user-centric view on this rights structures. An appropriate representation of the access rights should ease the detection of security gaps in a system.

Zusammenfassung

Der Bedarf an Sicherheit in der Informationstechnologie steigt. Dabei gilt es jedoch eine Reihe verschiedener Sicherheitsaspekte zu beachten. Einer davon ist zweifelsfrei die interne Sicherheit, die fälschlicherweise gerne als zweitrangig angesehen wird. Was nützt die beste Abschirmung eines firmeninternen Netzwerkes nach außen, wenn sich die Gefahrenquelle innerhalb befindet?

Dies ist zugleich der Ansatzpunkt des *SAT*-Projektes. Das *SAT2* System soll einen Administrator bei der Kontrolle der Zugriffsrechte unterstützen und damit einen Beitrag zur Gewährleistung der internen Sicherheit liefern. Da ein effizienter *Benutzer-zentrierter* Überblick über Benutzerberechtigungen mit den Standard-Werkzeugen des Windows 2000 Betriebssystems leider nicht möglich ist, wurde speziell für diesen Zweck das *Security Analysis Tool 2 (SAT2)* entwickelt.

Im Rahmen dieser Diplomarbeit wurde ein Prototyp zur Visualisierung der Rechtestrukturen, in Form eines *Microsoft Management Console (MMC) Snap-ins*, entwickelt, welcher eine *Benutzer-zentrierte Sicht* auf die Rechtestrukturen in einem Netzwerk ermöglicht. Durch eine prägnante Darstellung der Zugriffsrechte sollen dabei Sicherheitslücken in einem System leichter erkennbar werden.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich beim Erstellen dieser Diplomarbeit unterstützt haben.

- Bei meinem Betreuer o. Prof. Dr. Mühlbacher Jörg, dem Vorstand des Instituts für Informationsverarbeitung und Mikroprozessortechnik (FIM), für die Bereitstellung der Betriebsmittel und Räumlichkeiten, die zur Realisierung des *SAT2* Projektes benötigt wurden.

- Bei Dipl.-Ing. Hörmanseder Rudolf, dem geistigen „Ziehvater“ des *SAT* Projektes. Er hat viel Zeit und Energie in dieses Projekt investiert und stand mir jederzeit mit Rat und Tat zur Seite. Von ihm stammen viele der in dieser Diplomarbeit beschriebenen Ideen und Konzepte, auf welchen die Implementierung des *SAT* Systems aufbaut. Er hat damit maßgeblichen Anteil an dieser Arbeit.

- Bei MSR – Microsoft Research Cambridge (UK) für die finanzielle Unterstützung und das Interesse am *Security Analysis Tool 2 (SAT2)* – Projekt.

- Nicht zuletzt gilt mein besonderer Dank meiner Familie. Allen voran meiner Mutter Ingrid, meinem Vater Franz – insbesondere für die finanzielle Unterstützung – und natürlich auch meinem Bruder Wolfgang und meiner Freundin Kerstin, die in den letzten Monaten viel zu oft vernachlässigt wurden.

Danke!

Vorwort

Sicherheit von Computersystemen und Netzwerken erhält einen immer größeren Stellenwert auf dem Gebiet der Informationstechnologie. War noch vor einigen Jahren, als die IT noch in den Kinderschuhen steckte, die Devise „Freier Zugang zu allen Systemen, Daten und Informationen für Alle“, so stößt man heute auf das exakte Gegenteil. Ausgeklügelte Sicherheitssysteme verwehren – leider nicht nur – unbefugten Benutzern den Zutritt zu beinahe allen Computersystemen und Netzwerken bzw. den Zugriff auf Informationen und Daten. Betrachtet man allerdings die ständig steigende Zahl der Attacken sogenannter „*Hacker*“, können derartig strenge Sicherheitsbestimmungen nur befürwortet werden.

Bei der Entwicklung ihrer Betriebssysteme haben Firmen wie Novell oder Microsoft von Beginn an versucht, seitens des Betriebssystems einen Teil zum Sicherheitsmanagement beizutragen. Auf Grund der ständigen Erweiterungen erreichte man jedoch schnell einen Punkt, an dem alleine die Verwaltung der Benutzerberechtigungen zu einer äußerst komplexen Aufgabenstellung wuchs. Dies war unter anderem der Anlass zur Initiierung des *Projektes SAT* unter der Leitung von Hrn. Dipl.-Ing. Rudolf Hörmanseder, der zusammen mit Hrn. Kurt Hanner die erste Version des *Security Analysis Tools (SAT)* für *Microsoft Windows NT* entwickelte, welches im November 1999 fertiggestellt wurde und frei im Internet verfügbar ist [Sat]. Da der Fortschritt nicht aufzuhalten ist und Innovationen auf keinem Gebiet so schnell altern als auf dem der IT, ständig neue Technologien entwickelt oder verbessert werden, ist es nicht weiter verwunderlich, dass auch dieses Sicherheitsanalyse-Tool einer Weiterentwicklung bedarf.

Mit der Markteinführung der *Microsoft Windows 2000* Betriebssysteme haben sich einerseits alt bewährte Sicherheitskonzepte – wie z.B. das *NT Dateisystem (NTFS)* – geändert, andererseits wurden neue Technologien – wie z.B. *Active Directory Services (ADS)* – vorgestellt. Das *Security Analysis Tool 2 (SAT 2)* soll dem Anwender die Möglichkeit bieten, diese Neuerungen im Zuge der Sicherheitsanalyse auszuwerten.

Das Projekt *SAT* wird von Microsoft Research in Cambridge (UK) finanziell gefördert.

Inhaltsverzeichnis

1 Das SAT Projekt

Der Sicherheit von Computer-Netzwerken wird in den letzten Jahren immer größere Bedeutung zugewiesen. Nicht zuletzt deshalb, weil sich Angriffe von *Hackern* auf Computer-Systeme oder durch Viren verursachte Systemausfälle hervorragend für Sensationsmeldungen in den Medien eignen. Abgesehen davon ist der Schutz gegen solche Angriffe nur ein Teilaspekt der Computersicherheit. Denn Sicherheit in Computernetzwerken bedeutet weitaus mehr, als ein Netzwerk durch den Einsatz von *Firewalls* gegen Attacken von außen abzusichern, und setzt sich nicht nur aus Schlagworten wie: Antiviren-Programme, Datenverschlüsselung oder Authentifizierung zusammen.

Bedenkt man die Tatsache, dass mehr als die Hälfte aller Attacken auf Computer-Netzwerke von „Innen“ – d.h. der Angreifer hat direkten Zugang zum Netzwerk – erfolgen, so ist das Hauptaugenmerk vermutlich auf die „Interne Sicherheit“ zu legen. Vor allem beim Einsatz von Technologien wie den *Active Directory Services (ADS)*, welche die Möglichkeit bieten, alle nur erdenklichen Informationen über ein Unternehmen in einem zentralen Verzeichnis zu speichern, ist es besonders wichtig, diese Daten vor unberechtigtem Zugriff zu schützen.

Einen Ansatzpunkt zur Gewährleistung der internen Sicherheit bietet die korrekte Autorisierung durch eine konsistente Verwaltung von Benutzerberechtigungen, welche in den Aufgabenbereich des Netzwerk-Administrators fällt. Dass diese Aufgabe zu einer keineswegs trivialen werden kann, liegt an der hohen Komplexität der Aufgabenstellung, welche einerseits von der Größe des Netzwerkes und andererseits von der Anzahl der Benutzer abhängig ist. Zusätzlich erhöht der erweiterte Funktionsumfang von Betriebssystemen, z.B. durch verbesserte Möglichkeiten der Granularität der Rechtevergabe, diese Komplexität weiter.

Das *SAT-Team* hat sich zur Aufgabe gemacht, den Administrator bei der Verwaltung der Benutzerberechtigungen zu unterstützen, indem ihm eine Sicht auf Ressourcen in einem Netzwerk geboten wird, die in dieser Art und Weise von Standard-Werkzeugen zur Computer- oder Netzwerk-Verwaltung nicht unterstützt wird.

Dieses erste Kapitel soll als Einleitung dienen und dem Leser gewissermaßen Grundkenntnisse über das *SAT* Projekt vermitteln. In diesem Zusammenhang soll deutlich gemacht werden, mit welchen Schwierigkeiten sich ein Netzwerk-Administrator konfrontiert sieht, welche Ziele mit dem *SAT* Projekt verfolgt werden und welchen Beitrag das *Security Analysis Tool (SAT)* zur Sicherheit in Computer-Netzwerken leisten möchte.

1.1 Motivation

Obwohl sich die Administration eines Computers oder eines Computer-Netzwerkes in den letzten Jahren in viele verschiedene Richtungen entwickelt hat, ist eine der Hauptaufgaben eines Netzwerk-Administrators noch immer die Verwaltung des Zugriffs und damit der Zugriffsberechtigungen auf Netzwerk-Ressourcen. Dieser Zugriff auf beispielsweise Dateien, Verzeichnisse oder Drucker, kann durch die Vergabe von Benutzerberechtigungen geregelt werden. Oder anders formuliert: Netzwerk-Ressourcen können durch (korrekte) Vergabe von Benutzerberechtigungen vor unerlaubtem Zugriff geschützt werden.

Betrachtet man in diesem Zusammenhang das Dateisystem von *Microsoft Windows NT/2000/XP* Systemen – das sogenannte *NT Dateisystem (NTFS)* – so stellen Dateien und Verzeichnisse spezielle Objekte dar, auf die ein Benutzer, abhängig von seinen Berechtigungen, Zugriff haben kann. Die Rechtevergabe erfolgt demnach auf Objektebene, wobei so für jedes Objekt festgelegt wird, welcher Benutzer in welcher Weise auf das Objekt zugreifen darf. Zu diesem Zweck stellen *Microsoft Windows NT/2000/XP* Betriebssysteme verschiedene Standard-Tools, z.B. den *Windows-Explorer* zur Verfügung. Diese bieten dem Administrator eine Möglichkeit, die Zugriffsberechtigungen schnell und relativ einfach – jedoch nur für einzelne Objekte bzw. Gruppen von Objekten – festzulegen oder zu verändern. Ähnlich verhält sich dies beim Überprüfen oder Betrachten der Berechtigungen. Die Standard-Tools der Windows Betriebssysteme bieten dem Administrator leider nur eine – im Folgenden als *Objekt-zentrierte Sicht* bezeichnete – Sicht auf die Benutzerberechtigungen eines Objektes. D.h., dass Rechte, wie sie mit Standard-Werkzeugen vergeben, also per Objekt, auch nur per Objekt betrachten wer-

den können. Es besteht daher lediglich die Möglichkeit, für ein ausgewähltes Objekt festzustellen, mit welchen Rechten welche Benutzer darauf zugreifen dürfen.

Als Beispiel für den Aufwand, den eine derartige Zugriffskontrolle mit sich bringt, stelle man sich einen kleinen *File Server* vor, auf welchem „nur“ einige tausend Dateien gespeichert sind. Um die Zugriffsrechte eines bestimmten Benutzers auf dem Server überprüfen zu können, müsste der Administrator theoretisch jede einzelne der abgelegten Dateien kontrollieren. Und selbst bei einer geringen Zahl von Dateien ist alleine der Zeitaufwand, den die Kontrolle in Anspruch nimmt, enorm. Zusätzlich wird diese Aufgabe durch den Vererbungsmechanismus des *NT Dateisystems* erschwert, der auf Grund seiner Komplexität manchmal sogar erfahrene Netzwerk-Administratoren den Überblick verlieren lässt.

Wünschenswert wäre demnach eine *Benutzer-zentrierte Sicht*, eine Auflistung aller Objekte, auf die ein bestimmter Benutzer Rechte hat, um sich quasi einen Gesamtüberblick über dessen Berechtigungen verschaffen zu können. Und genau dieser „Wunsch“ war Anlass, das *SAT* Projekt ins Leben zu rufen. Mit dem Ziel, die Mängel in der Verwaltung der Benutzerberechtigungen seitens des Betriebssystems auszumerzen und durch die Entwicklung einer Analyse-Applikation – des *Security Analysis Tools (SAT)* – den Administrator bei seiner Arbeit zu unterstützen.

1.2 Die Geschichte des Security Analysis Tools (SAT)

Das Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM), der Johannes Kepler Universität Linz, an dem das *Security Analysis Tool (SAT)* entwickelt wird, verwendet seit der Markteinführung von *Microsoft Windows NT 3.1* Microsoft-Betriebssysteme zur Organisation des Institutseigenen Computer-Netzwerkes.
Aus dieser Zeit stammt auch die Idee zum *SAT* Projekt. Im Jahre 1995 wurden, auf Grund der mangelnden Funktionalität in Bezug auf die Verwaltung von Benutzerberechtigungen des Microsoft Betriebssystems, erste Überlegungen angestellt, einem Administrator eine völlig neue *Benutzer-zentrierte Sicht* auf die Rechtestrukturen in einem Netzwerk zu bieten.

Die Grundidee zu diesem Projekt stammt von Hrn. Dipl.-Ing. Rudolf Hörmanseder, selbst Netzwerk-Administrator und Mitarbeiter des oben genannten Institutes, und kann in [Hoe1] nachgelesen werden. Gemeinsam mit dem damaligen Studenten Kurt Hanner entwickelte Dipl.-Ing. Hörmanseder, der mit der Projektleitung beauftragt wurde, ein Basiskonzept, woraus die Implementierung des *SAT* Prototyps entstand.

Kurt Hanner schilderte den Anlass zur Initiierung des Projektes, im Vorwort seiner 1998 veröffentlichten Diplomarbeit [Han] („Analyse und Archivierung von Benutzerberechtigungen in einem Windows NT Netzwerk"), wie folgt:
„*Trotz aller Leistungsfähigkeit des neuen Systems, hat man Schwächen gegenüber dem älteren Novell System ausgemacht. Windows NT bietet im Gegensatz zum Novell System nur sehr eingeschränkte Möglichkeiten bei der Verwaltung von Benutzerberechtigungen.*"

Im November 1999 wurde die Implementierung des Sicherheitsanalyse-Tools fertiggestellt und die Version *SAT 1.0 Beta* der Öffentlichkeit vorgestellt. Seither kann *SAT* vom FIM via Internet (http://www.fim.uni-linz.ac.at/sat) kostenlos bezogen werden.

1.3 Funktionalität und Architektur von SAT 1.0

Im folgenden Abschnitt soll zusammenfassend die Funktionalität und Architektur des *Security Analysis Tools 1.0 (SAT 1.0)* beschrieben werden. Für Interessenten sei an dieser Stelle nochmals auf die Diplomarbeit von Hrn. Kurt Hanner [Han] verwiesen, in dessen Arbeit die detaillierte Beschreibung des ursprünglichen Sicherheitsanalyse-Tools nachgelesen werden kann.

Das erklärte Ziel, welches mit dem *SAT* Projekt verfolgt wird ist, den Administrator bei der Verwaltung der Benutzerberechtigungen zu unterstützen. Zu diesem Zweck wurde eine Applikation entwickelt, die eine *Benutzer-zentrierte Sicht* auf die Rechtestrukturen in einem Netzwerk ermöglicht. Ausgehend von der Fragestellung: „Wo hat ein bestimmter Benutzer Rechte?", soll dem Administrator eine Auflistung all jener Objekte geboten werden, auf welche der ausgewählte Benutzer Zugriffsrechte besitzt.

Um diese Ergebnisse in angemessener Zeit und in einer übersichtlichen und aussagekräftigen Form darstellen zu können, bestand die Notwendigkeit, die Applikation in zwei Teile aufzuspalten.

Ein Teil – der Datensammler – soll alle relevanten Informationen über die Sicherheitseinstellungen im Netzwerk sammeln und in einer Datenbank speichern. Der andere Teil – eine Visualisierungs-Applikation – übernimmt die gespeicherten Daten und sorgt für eine übersichtliche Darstellung der Analyse-Ergebnisse.

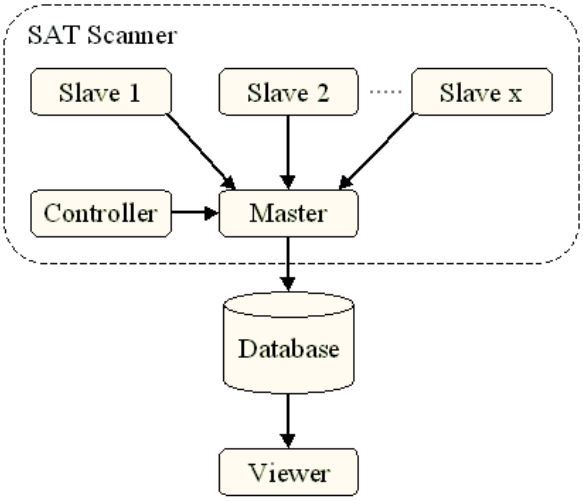


Abbildung 1.1: Architektur SAT 1.0

Abbildung 1.1 (entnommen aus [Hoe1]) zeigt die Architektur des *Security Analysis Tools 1.0 (SAT 1.0)*, wodurch verdeutlicht werden soll, aus welchen Teilen sich die Applikation zusammensetzt und wie die einzelnen Komponenten miteinander in Verbindung stehen.

Der ***Daten-*** oder ***Rechte-Sammler*** setzt sich aus folgenden drei Komponenten zusammen:

- ***Master-Komponente:***
  Ist verantwortlich für die Verwaltung der *Slaves* und das Speichern, der von den *Slaves* angelieferten Informationen in der zentralen Datenbank.

- ***Slave-Komponente:***
  Liest die Sicherheitseinstellungen eines zu untersuchenden Computers aus und sendet die gesammelten Daten an die *Master*-Komponente. *Slaves* können parallel auf verschiedenen Rechnern im Netzwerk eingesetzt werden.

- ***Controller-Komponente:***
  Fungiert als Schnittstelle zwischen Benutzer und Rechte-Sammler und ist zuständig für das Konfigurieren und Initiieren eines Analyselaufes.

Die ***Visualisierungs-Applikation*** stellt das Kernstück der Applikation dar. Dieser Teil des *Security Analysis Tools 1.0 (SAT 1.0)* sorgt einerseits für die Auswertung der in der Datenbank gespeicherten Sicherheitseinstellungen, und andererseits für eine visuelle Aufbereitung der analysierten Daten bzw. für das Generieren der *Benutzer-zentrierten Sicht* auf die Benutzerberechtigungen.

Einsatzbereich und Laufzeitumgebung:
Das System wurde – den damaligen Gegebenheiten angepasst – zur Analyse der Benutzerberechtigungen in *Microsoft Windows NT* Netzwerken entwickelt. Im Speziellen bedeutet dies, dass die Applikation auf *Windows NT 3.x und 4.0* Rechnern (sowohl *Server*- als auch *Workstation*-Version) eingesetzt werden kann. Die Visualisierungs-Applikation läuft zusätzlich auf *Windows 95/98* Rechnern. Unterstützt wird die Analyse von Benutzerberechtigungen im *NT Dateisystem* (*NTFS*), wobei die gesammelten Sicherheitseinstellungen in einer *Microsoft Access* Datenbank gespeichert werden.

1.4 Weiterentwicklung von SAT 1.0

Wie eingangs erwähnt stammt die Idee zum *SAT* Projekt bereits aus 1995, woraufhin die erste Version des *Security Analysis Tools (SAT)* im Jahre 1999 veröffentlicht wurde. Für Software-Produkte ist dies ein beträchtliches Alter, wenn man bedenkt, dass sich alleine am Betriebssystem-Sektor in den letzten Jahren vieles verändert hat.

Microsoft entwickelte neue Betriebssysteme, die eine Vielzahl an Änderungen und Neuerungen mit sich brachten. Mit der Markteinführung der *Windows 2000* Betriebs-

systeme, insbesondere der *Server*-Familie, wurde nicht nur das *Active Directory (AD)* vorgestellt, sondern auch eine neue Version des *NT Dateisystems* (*NTFS 5*) veröffentlicht. Darüber hinaus kommen jede Menge weiterer Sicherheitskonzepte zum Einsatz.
Auch in Bezug auf die Computer- und Netzwerkverwaltung versucht Microsoft neue Maßstäbe zu setzen. Durch den Einsatz der *Microsoft Management Console (MMC)* sollen diverse Administrations-Werkzeuge in einer gemeinsamen Benutzeroberfläche verfügbar sein.

Diese Neuerungen gaben den Anlass zur Weiterentwicklung des Sicherheitsanalyse-Tools. In einer völlig neuen Version, basierend auf vielen der ursprünglichen Ideen, soll dem Administrator eines *Windows 2000* Netzwerkes die Möglichkeit geboten werden, nicht nur Benutzerberechtigungen im *NTFS*, sondern auch im *Active Directory* und in der *Windows-Registry*, analysieren zu können. Weiters wurden sowohl das Datenbankkonzept überarbeitet, als auch die Visualisierungs-Applikation neu entwickelt, die nunmehr als sogenanntes *Snap-in* für die *Microsoft Management Console (MMC)* existiert.

In Kapitel 3 dieser Arbeit erfolgt eine detaillierte Beschreibung des *Security Analysis Tools 2 (SAT 2)*. Dabei liegt der Schwerpunkt eindeutig im Bereich der Analyse der Berechtigungen und der Visualisierung – dem Aufgabenbereich des Autors dieser Arbeit.

2 Die Microsoft Management Console

Das Einsatzgebiet der *Microsoft Management Console (MMC)* liegt typischerweise im Bereich der Computer-, Netzwerk- bzw. Systemverwaltung. Die MMC selbst bietet jedoch – wie die Bezeichnung vielleicht vermuten ließe – keine Management-Funktionalität, vielmehr stellt sie „nur“ eine einheitliche Benutzeroberfläche zur Einbindung der unterschiedlichsten Tools für die Verwaltung von Windows-basierten Systemen oder Netzwerken, der sogenannten *Snap-ins*, dar. Die große Akzeptanz der MMC, die Vielzahl der verfügbaren *Snap-ins* und das Ziel von Microsoft, die MMC künftig als Basis für alle Verwaltungsaufgaben in Windows Betriebssystemen zu verwenden, dienten als Entscheidungsgrundlage für einen Einsatz der *Management Console* im Projekt *SAT 2*.

In folgendem Kapitel sollen dem Leser Grundkenntnisse über die *Microsoft Management Console (MMC)* vermittelt werden. Darüber hinaus werden grundlegende Konzepte, Funktionalität und verfügbare Steuerelemente beschrieben, sowie wesentliche Aspekte, welche bei der Entwicklung eines *Snap-ins* beachtet werden sollten, erläutert.

An dieser Stelle soll auf die bis dato einzige zur Verfügung stehende Literatur – in Bezug auf die *Microsoft Management Console (MMC)* – verwiesen werden. [Mmc1] bzw. [Mmc2] dienten als Basislektüre zur Erstellung dieses Kapitels und können dem Leser als Nachschlagewerk für die hier beschriebenen Fakten aber auch für nicht angeführte (meist implementierungstechnische) Details dienen.

2.1 Die (kurze) Geschichte der MMC

Was würde sich bei der Vorstellung eines Produkts besser als Einleitung eignen, als ein geschichtlicher Rückblick über die Entstehung und Entwicklung desselben? Deshalb wird im folgenden Abschnitt die relativ kurze Geschichte der MMC, frei nach den Worten ihres Erfinders Tony Romano [Rom], erzählt:

Im Juni 1996 beendeten Mike Miller und Tony Romano die Entwicklung des *Network Control Panel (NCP)* für *Windows NT 4.0* und kurz darauf begannen für Tony Romano die Arbeiten an einem neuen Projekt.

Er sollte eine Administrationsoberfläche für den *Routing and Remote Access Service (RRAS)* entwerfen und implementieren. Ursprünglich war geplant, den *NCP* so zu ändern, dass spezielle Eigenschaftenseiten aufgerufen werden können, falls auf dem zu administrierenden Computer der *RRAS* installiert ist. Der Grund dafür war naheliegend: die *RRAS* Administration sollte nach dem selben Schema ablaufen wie die Konfiguration eines Netzwerks. Zusätzlich sollte außenstehenden Entwicklern die Möglichkeit geboten werden, eigene Eigenschaftenseiten für benutzerdefinierte *Routing Protokolle*, welche den *RRAS* erweitern, in das Tool einbinden zu können.

Im weiteren Verlauf der Anforderungsanalyse bemerkte Tony Romano schnell, dass die Vielzahl an Erweiterungen, die erforderlich waren, im Vergleich zu den Möglichkeiten des *NCP*, in keinem Verhältnis zueinander standen. Er entschied deshalb, eine andere Richtung einzuschlagen.

Im Oktober 1996 wurde der Grundstein für die heute weit verbreitete *Microsoft Management Console (MMC)* gesetzt. Zusammen mit Wayne Scott entwickelte Tony Romano den ersten, äußerst einfachen Prototypen der MMC. Das Grundgerüst stellte eine erweiterbare, auf dem *Windows Explorer* basierende Benutzerschnittstelle dar. Auch die Ziele hatte man bereits klar vor Augen. Möglichst viele Administrations-Tools sollten auf diesem Modell aufbauen können, also in einer einheitlichen Benutzeroberfläche einsetzbar sein und somit zu beliebig kombinierbaren, wiederverwendbaren Komponenten werden.

Nach Fertigstellung des Prototyps wurde dieser zuerst dem *RRAS*-Team und anschließend Dan Plastina vorgeführt, der damals für die Entwicklung neuer Tools für Windows NT verantwortlich war und ähnliche Ideen hatte.

Ab diesem Zeitpunkt entwickelte Tony Romano mit einem eigenen Projektteam die nächste Generation der Windows NT Verwaltung – die MMC war geboren.

2.2 Einführung

Vor einigen Jahren musste sich ein Administrator „nur“ mit einfachen Aufgaben wie z.B. Prüfung der Integrität einer Festplatte oder dem Sicherstellen von Modemverbindungen bzw. in Computernetzwerken mit der korrekten Freigabe von Dateien, Netzwerkdruckern und dgl. auseinandersetzen.
Heutzutage verfügt beinahe jeder Computer über einen Internetzugang, die Zahl der finanziell erschwinglichen Peripheriegeräte (Videokameras, Digitalkameras, DVD Laufwerke, usw.) wird immer größer und speziell im Software-Bereich steigt sowohl die Funktionalität als auch die Komplexität der Applikationen. All diese Innovationen fordern zumindest zeitweilig die Aufmerksamkeit des Administrators.
Die Administration eines einzelnen Computers zu unterstützen, reicht heute bei weitem nicht mehr aus. Ein Administrator muss heute ein gesamtes Netzwerk effizient und zentral administrieren können. Vor allem im Netzwerk-Bereich steigt der Administrationsaufwand auf Grund neuer Technologien und einer ständigen Vergrößerung der Netze kontinuierlich an. Vernetzte Systeme können z.B. *Services* zur Verfügung stellen, die entweder lokal oder unter Verwendung des Internet auch global zum Einsatz kommen. Und obwohl viele Aufgaben eines Netzwerk-Administrators äquivalent zu den Aufgaben in der Verwaltung eines einzelnen Computers sind, liegen im Bereich der Netzwerk- und Server-Administration und auch der Client-Administration eine Menge zusätzlicher Anforderung verborgen.

Einige der Probleme bzw. Schwierigkeiten, mit denen sich ein Administrator eines Computer-Netzwerkes aber auch eines einzelnen Computers konfrontiert sieht, seien im Folgenden kurz erwähnt:

- ***Administrations-Tools sind oft schwer zu finden:***
  Die große Zahl der heute verfügbaren Tools zur Computer- oder Netzwerk-Verwaltung, trägt ihren Teil zu den Schwierigkeiten auf der Suche nach einer passenden Applikation bei. Sei dies die Suche nach dem richtigen Treiber oder einem (Steuerungs-)Programm für ein spezielles Peripheriegerät oder einer Applikation zur Durchführung einer bestimmten Verwaltungsaufgabe. Aber auch das Auffinden eines Programms im „Start-Menü“ des *Windows* Betriebssystems kann sich als durchaus aufwendig gestalten.

- ***Jedes Tool hat seine eigene Benutzeroberfläche:***

  Typischerweise verwendet ein *Server*-Administrator eine Menge verschiedener Tools, um seinen *Server* zu administrieren. Mit Hilfe verschiedener Applikationen kann er u.a. Datenbanken, Drucker, Netzwerkverbindungen, Benutzerkonten, Hardware oder automatische Software-Installationen kontrollieren. Viele dieser Applikationen besitzen meist unterschiedliche Benutzeroberflächen und erschweren dadurch die Arbeit des Administrators.

- ***Mangelnde Flexibilität der meisten Tools:***

  Die meisten Administrations-Tools sind weder an das Wissen noch an die Fähigkeiten des Administrators anpassbar. So scheitern unerfahrene Administratoren oft an der Komplexität eines Tools, wobei im Gegensatz dazu erfahrene Administratoren oft durch langwierige, schrittweise – dafür einfache und verständliche – Durchführung routinemäßiger Aufgaben durch das Tool in ihrer Arbeitsweise eingeschränkt werden.

- ***Tools sind zumeist „Rechner-zentriert“:***

  Viele Standard-Tools (von Windows NT/2000) ermöglichen ausschließlich eine Administration des Rechners auf dem sie installiert wurden bzw. ausgeführt werden und sind demnach nicht *remote*-fähig (z.B. *Disk-Administrator*). Andere Tools wiederum, wie beispielsweise der *Event-Viewer*, sind zwar *remote*-fähig müssen aber für jeden zu administrierenden Rechner jeweils einmal gestartet werden und erfordern dadurch ein ineffizientes hin- und herschalten zwischen den einzelnen Programm-Instanzen bzw. *Event-Logs*. Im Vergleich dazu bietet der *NT4 Server-Manager* zwar Möglichkeiten zur Verwaltung aller Computer einer Domain – allerdings ohne Einschränkung – verfügt aber in Hinblick auf deren Administration über einen minimalen Funktionsumfang.

All diese Gesichtspunkte bringen einerseits eine zu lange Einlernphase für „Neulinge“ und andererseits teilweise Frustration für routinierte System-Administratoren mit sich. Wünschenswert wäre demnach eine Applikation mit einheitlicher Benutzeroberfläche, die dem Administrator ein effizientes Durchführen von (möglichst allen) Verwaltungsaufgaben ermöglicht.

Die *Microsoft Management Console (MMC)* leistet einen wesentlichen Beitrag zur Lösung der zuvor angesprochenen Probleme. Das verfolgte Ziel ist, durch den verbreiteten Einsatz der MMC den Administrationsaufwand Windows-basierter Systeme zu minimieren. Zu diesem Zweck wird ein einfaches, konsistentes, erweiterbares und integriertes *User-Interface*, aber auch ein Administrations-Modell, zur Verfügung gestellt.

2.3 Konsolen und Snap-ins

Die *Microsoft Management Console (MMC)* wurde entwickelt, um Probleme in der Computer- und Netzwerk-Administration zu reduzieren und den damit verbundenen Aufwand zu minimieren.
Der Einsatz der MMC eröffnet dem Benutzer die Möglichkeiten zum „Öffnen“, „Speichern“ oder „Erzeugen“ von Administrations-Tools (sog. *MMC Konsolen*) zur Verwaltung von Hardware, Software und Netzwerk-Komponenten in einem *Windows* System.

Die MMC selbst verfügt über keine administrative Funktionalität, sie agiert lediglich als *Host* für administrative Komponenten. Diese Komponenten werden als sogenannte *Snap-ins* bezeichnet und spezifizieren die Funktionalität einer *MMC Konsole*. Ein *Snap-in* bezeichnet also ein speziell für die MMC entwickeltes Administrations-Tool. Darüber hinaus können einer Konsole sowohl *ActiveX Controls*, als auch Links zu Web-Seiten, Ordner oder sogenannte *Taskpad Views* hinzugefügt werden.

Die MMC basiert auf einem *Multiple Document Interface (MDI)* Konzept, wobei jedes geöffnete Fenster eine eigene „*View*“ repräsentiert. Die Fenster stehen in keinerlei Abhängigkeit zueinander und können ein oder mehrere *Snap-ins* enthalten.

Abbildung 2.1 zeigt eine MMC Konsole, in der das *SAT2 Snap-in* zur Visualisierung von Benutzerberechtigungen geöffnet ist, welches im Rahmen dieser Diplomarbeit entwickelt wurde.

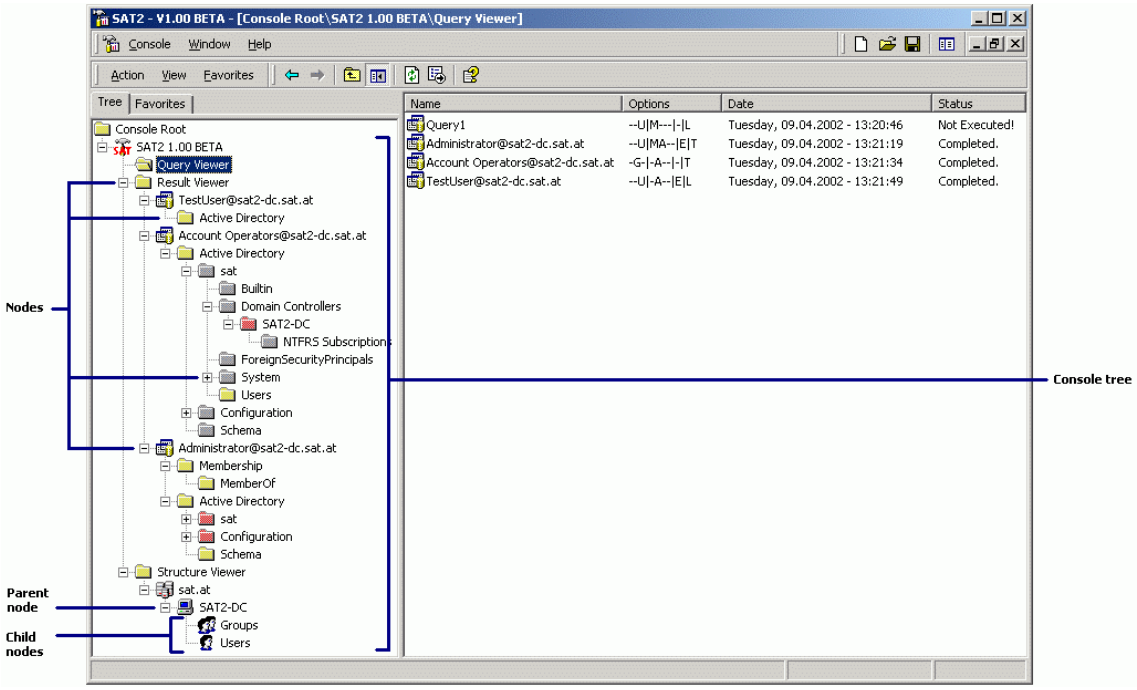


Abbildung 2.1: Microsoft Management Console (MMC)

Die graphische Oberfläche erinnert stark an die des *Windows-Explorers*, genauso wie das grundsätzliche Verhalten der Konsole bei Benutzer-Eingaben (z.B. beim Maus-Klick auf ein dargestelltes Objekt).

Unter Verwendung von Abbildung 2.1 soll hier zu Beginn das „Knoten-Konzept“ der MMC verdeutlicht werden, das die Basis zum Verständnis des folgenden Kapitels bildet.

Die linke Seite der Konsole enthält eine Baumstruktur (*Console Tree*), die gewissermaßen eine hierarchische Anordnung von Knoten (*Nodes*) – den Zweigen des Baumes – repräsentiert.

Als Vaterknoten (*Parent Nodes*) werden jene Knoten bezeichnet, für die im Baum eine Hierarchie-Ebene tiefer, weitere Knoten existieren. Diese Knoten wiederum werden Söhne (*Child Nodes*) genannt.

2.3.1 MMC Konsolen

Die MMC kann in zwei verschiedenen Modi betrieben werden: im Autorenmodus (*Author Mode*) und im Benutzermodus (*User Mode*).

Im Autorenmodus erhält der Benutzer vollständigen Zugriff auf die gesamte MMC Funktionalität, einschließlich der Möglichkeit, *Snap-ins* hinzuzufügen oder zu entfernen, neue Fenster zu erstellen, *Taskpad Views* und *Tasks* zu erzeugen und alle Teile der Konsolenstruktur anzuzeigen. Darüber hinaus besteht die Möglichkeit, über verschiedene Dialoge das *Look-and-Feel* der Konsole an seine eigenen Bedürfnisse anzupassen.

Für den Benutzermodus stehen drei unterschiedliche Zugriffsoptionen zur Auswahl, die je nach Option den Zugriff des Benutzers auf die MMC Funktionalität einschränken:

- ***Benutzermodus – Vollzugriff:***
  Ermöglicht dem Benutzer den vollständigen Zugriff auf alle Befehle zur Fensterverwaltung und auf die bereitgestellte Konsolenstruktur. Es ist dem Benutzer jedoch nicht möglich, *Snap-ins* hinzuzufügen oder zu entfernen oder die Konsoleneigenschaften zu ändern.

- ***Benutzermodus – beschränkter Zugriff, mehrere Fenster:***
  Ermöglicht dem Benutzer nur den Zugriff auf die Bereiche der Konsolenstruktur, die sichtbar waren, als die Konsole gespeichert wurde. Der Benutzer kann neue Fenster erstellen, jedoch keine vorhandenen Fenster schließen.

- ***Benutzermodus – beschränkter Zugriff, Einzelfenster:***
  Ermöglicht dem Benutzer nur den Zugriff auf die Bereiche der Konsolenstruktur, die sichtbar waren, als die Konsole gespeichert wurde. Es ist dem Benutzer nicht möglich, neue Fenster zu öffnen.

Einem System-Administrator wird mit dem Konsolen-Konzept die Möglichkeit geboten, eine neue Konsole zu erstellen, die sich aus einem oder mehreren *Snap-ins* zusammensetzt, und diese (als *.msc* Datei) zu speichern. Darüber hinaus können auf diese Weise mehrere ausgewählte Computer eines Netzwerkes von einer „zentralen Stelle“ aus verwaltet werden.

Abbildung 2.2 zeigt eine benutzerdefinierte MMC Konsole, in der verschiedene *Snap-ins*, welche zur Administration eines Computer-Netzwerkes eingesetzt werden können,

geladen wurden. Um die Möglichkeit aufzuzeigen, mehrere Computer zu administrieren, wurden alle *Snap-ins* jeweils zweimal geöffnet. Einmal, zur Verwaltung des Rechners, auf dem die MMC Konsole erstellt wurde („*Local*“) und ein zweites Mal zur Verwaltung eines Rechners im Netzwerk („*P120*“).

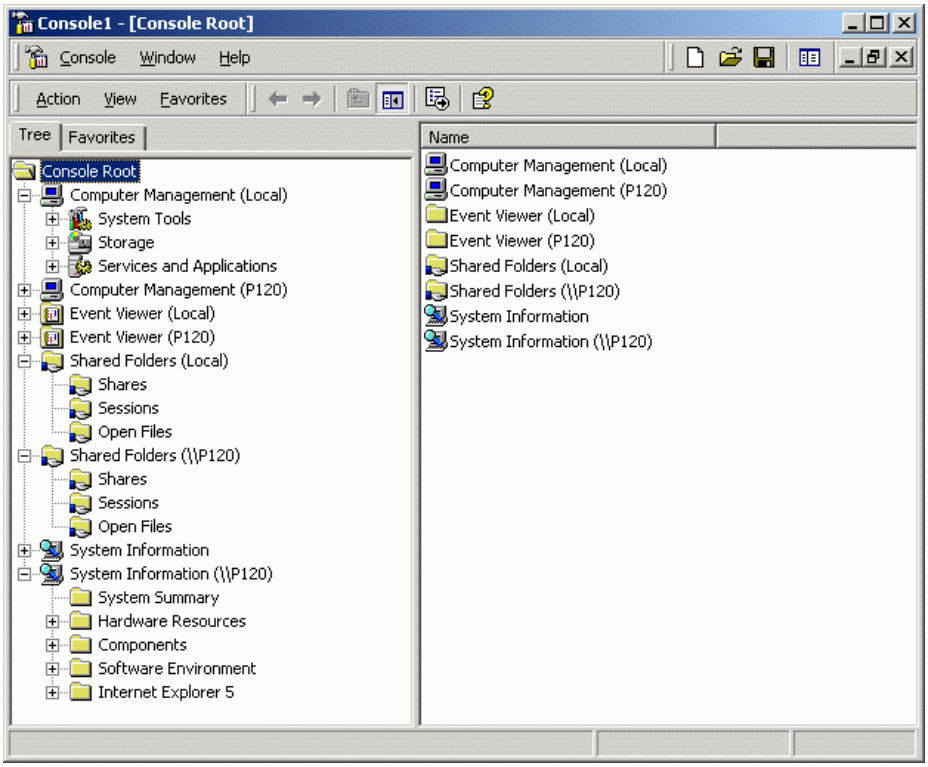


Abbildung 2.2: Zentrale Administration

Die aus dem Konsolen-Konzept resultierenden Vorteile liegen auf der Hand. Vordergründig ist zweifellos die Möglichkeit zur zentralen Administration eines Netzwerkes. Ferner ist einerseits die Wiederverwendbarkeit gewährleistet, ohne jedes Mal aufs Neue die *Snap-ins* zusammenstellen zu müssen, andererseits können diese Tools leicht unter den Administratoren ausgetauscht werden. Ein weiterer Vorteil den dieses Konzept mit sich bringt ist, dass auf diese Weise für unterschiedliche administrative Aufgaben spezifische Tools erstellt werden können, die exakt über die benötigte Funktionalität verfügen und quasi als alleinstehende Applikationen zum Einsatz kommen.

Jeder Benutzer – Administrator oder nicht – kann die MMC im Autorenmodus bedienen und hat damit die Möglichkeit, neue Konsolen als Kombination von bestehenden *Snap-ins* zu erzeugen. Durch Hinzufügen der gewünschten *Snap-ins*, Spezifizieren der Anzei-

ge-Optionen und anschließendes Speichern der Konsole (als *.msc* Datei) können so benutzerdefinierte und aufgabenspezifische Tools erstellt werden. Auf diesem Weg kann ein Tool an die Fähigkeiten des Benutzers angepasst und natürlich auch die Art und Weise, wie er auf welche Daten zugreifen und sie manipulieren soll, kontrolliert werden.

2.3.2 MMC Snap-ins

Die *Microsoft Management Console (MMC)* verfügt, wie bereits erwähnt, selbst über keinerlei Management-Funktionalität, sie stellt lediglich eine gemeinsame Benutzeroberfläche für administrative Komponenten – die sogenannten *MMC Snap-ins* – zur Verfügung. Die Funktionalität einer *MMC Konsole* ist daher vollständig von der Funktionalität der *Snap-ins* abhängig, welche der Konsole hinzugefügt wurden.

Die Kommunikation zwischen MMC und *Snap-in* erfolgt über eine *Component Object Model (COM)* Schnittstelle, die sich natürlich zwischen den beiden Komponenten befindet (siehe Abbildung 2.3).

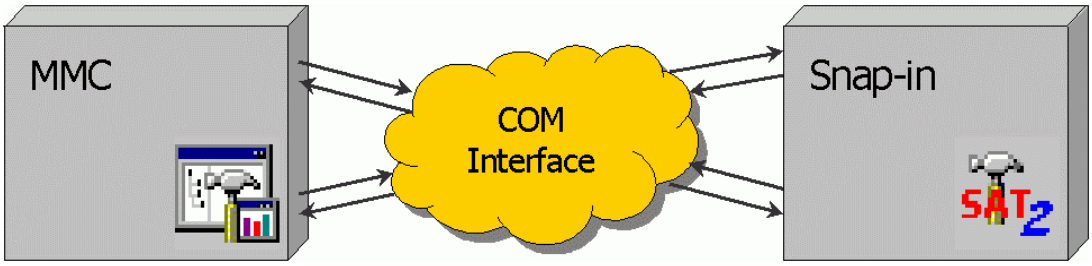


Abbildung 2.3: COM-Schnittstelle

Auf Grund dieser Architektur muss die MMC weder „wissen“, wie das *Snap-in* Verwaltungsaufgaben durchführt, noch welche Mechanismen zur Realisierung dieser Aufgaben eingesetzt werden. Dadurch wird auch von *Microsoft* unabhängigen Software-Herstellern die Möglichkeit geboten, *Snap-ins* zu entwickeln.

Grundsätzlich muss zwischen zwei Arten von *Snap-ins* unterschieden werden: den „Eigenständigen (*Stand-alone*) *Snap-ins*“ und den „Erweiterungs (*Extension*) *Snap-ins*“.

- ***Stand-alone Snap-ins:***
  Können als einziges in einer Konsole geladenes *Snap-in* ihre designierten Verwaltungsaufgaben durchführen.

- ***Extension Snap-ins:***
  Erweitern die Funktionalität eigenständiger *Snap-ins*. Sie können entweder ihre eigenen Knoten als Söhne der Knoten des eigenständigen *Snap-ins* hinzufügen oder einen vorhandenen Knoten durch (zusätzliche) Kontextmenüs, Eigenschaftenseiten, Symbolleisten bzw. Schaltflächen und *Taskpad*-Ansichten erweitern.

Darüber hinaus können *Snap-ins* eine Art Doppelrolle übernehmen. Sie werden als sogenannte ***Dual-mode Snap-ins*** bezeichnet. Auf diese Weise können sie einerseits als eigenständige *Snap-ins* und andererseits zur Erweiterung der Funktionalität eines anderen *Snap-ins* eingesetzt werden.

2.4 Der MMC Namespace

Der Begriff *Namespace* wird auf dem Gebiet der Informationstechnologie vielseitig verwendet und wird in [Msc1/S.1453] wie folgt definiert:
„Ein Satz eindeutiger Namen für Ressourcen oder Elemente, die in einer gemeinsam genutzten Computerumgebung verwendet werden. Die Namen in einem Namespace können in die Objekte aufgelöst werden, die sie repräsentieren.“

Im Zusammenhang mit der *Microsoft Management Console (MMC)* wird dieser Begriff folgendermaßen ausgelegt:
„Bei der MMC-Konsole wird der Namespace durch die Konsolenstruktur repräsentiert, die alle Snap-ins und Ressourcen anzeigt, die von der Konsole aus zugänglich sind“ ([Msc1/S.1453]).

Folgende Abbildung (Abbildung 2.4) soll verdeutlichen, wie sich der *MMC Namespace* zusammensetzt. Unterhalb der Standard-Symbolleisten befinden sich zwei Fenster. Das linke Fenster zeigt den *Console Tree* (siehe Kapitel 2.3), der alle *Snap-ins* enthält, aus denen sich das Administrations-Tool zusammensetzt und wird als *Scope Pane* bezeich-

net. Das Fenster auf der rechten Seite stellt den *Result Pane* dar und zeigt jeweils Details über den momentan in der Baumstruktur ausgewählten (markierten) Knoten. Zusammen bilden diese beiden Teile den *MMC Namespace*.

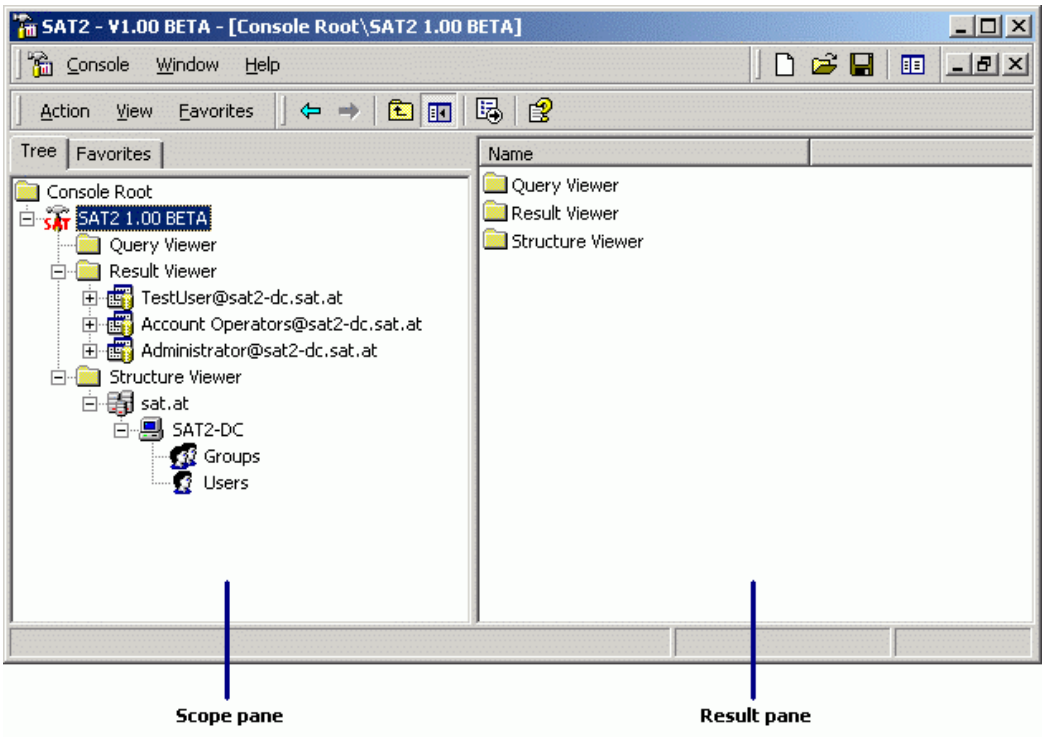


Abbildung 2.4: MMC Namespace

Das wichtigste Merkmal eines Knoten im *Namespace* ist sein Typ. Der Knotentyp legt fest, welches Objekt durch den Knoten repräsentiert werden soll. Diese Objekte können z.B. Benutzer, Computer, Domaineinträge in einer *DNS*-Datenbank und dgl. sein. Der Typ eines Knoten wird durch eine eindeutige ID, einen *Global Unique Identifier (GUID)* bestimmt. Anhand dieses *GUIDs* können die einzelnen Knoten im *Namespace* eindeutig identifiziert und so eventuelle Namenskonflikte vermieden werden. Um diese Eindeutigkeit gewährleisten zu können, dürfen die *GUIDs* nicht durch die MMC festgelegt, sondern müssen durch die einzelnen *Snap-ins* definiert werden.

2.4.1 Der Scope Pane

Der *Console Tree* setzt sich aus Knoten zusammen, die sowohl Container als auch spezielle Objekte repräsentieren können. Die Wurzel der Baumstruktur eines jeden *Standalone Snap-in* wird als *Static Node* bezeichnet, der als einziger Knoten von der MMC

verwaltet wird. Deshalb kann dieser Knoten, d.h. sein Name und Symbol, bereits beim Öffnen einer gespeicherten Konsole angezeigt werden, auch wenn das *Snap-in* selbst noch nicht (vollständig) geladen wurde.

Für das Erzeugen und Einfügen der Söhne des *Static Node* in die Baumstruktur des *Scope Pane* ist das *Snap-in* selbst verantwortlich, genau so wie für deren Verwaltung. Diese Knoten werden auch als dynamische Knoten bezeichnet, da die MMC nichts über ihre Existenz und Eigenschaften weiß, solange sie nicht in den Baum eingefügt wurden. Grundsätzlich werden die Söhne vom Vaterknoten erzeugt und erst beim Doppelklicken bzw. Expandieren (durch Mausklick auf das Plus-Symbol) in die Baumstruktur eingefügt.

Auf Grund dieser dynamischen Natur der Sohnknoten muss der MMC beim Einfügen eines Knotens mitgeteilt werden, ob dieser Söhne hat oder nicht. Diese Benachrichtigung muss an die MMC gesendet werden, um festlegen zu können, ob in der Baumstruktur vor dem Knoten ein Plus-Symbol angezeigt werden muss.

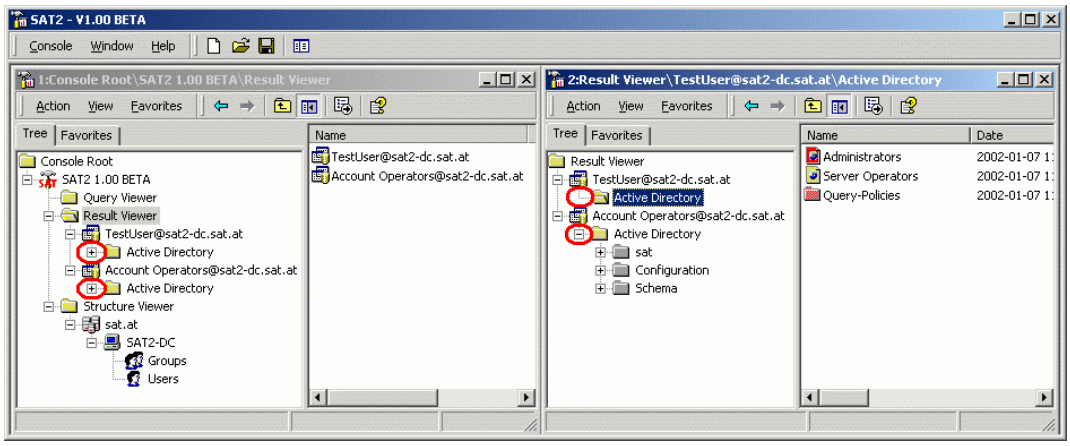


Abbildung 2.5: Anzeige der Plus-Symbole

Dies impliziert aber, dass bereits zum Zeitpunkt des Einfügens eines Knoten feststehen muss, ob er einen Endknoten im Baum repräsentiert und steht daher im Widerspruch zur bereits angesprochenen Dynamik. Die Eigenschaft Endknoten ist beim Erzeugen eines Knoten nicht immer einfach festzustellen, woraus eine teilweise inkorrekte Anzeige der Plus-Symbole resultiert. Eingeschränkt kann dieser Fehler nur durch eine Abschätzung der Wahrscheinlichkeit des Auftretens von Söhnen und der darauf basierenden Ent-

scheidung für oder gegen die Anzeige des Plus-Symbols werden. Teilweise wird das Plus-Symbol standardmäßig immer angezeigt, wie z.B. beim *ADSI-Edit Snap-in*, denn die MMC korrigiert falsch angezeigte Symbole automatisch nach der Selektion eines Knotens durch den Anwender.
Auch im *SAT2 Snap-in* ist eine korrekte Anzeige der Plus-Symbole nicht immer möglich (siehe Abbildung 2.5). Wird im Ordner „*Active Directory*“ kein Verzeichnisbaum zur Darstellung der Analyseergebnisse aufgebaut, sondern lediglich eine Liste von Ergebnissen generiert, wird fälschlicherweise das Plus-Symbol angezeigt. Dies resultiert aus den unterschiedlichen Darstellungsarten (Baum- und Listendarstellung), die ausgewählt werden können, und der Annahme, dass dem Anwender besser ein Plus-Symbol angezeigt wird, wo eigentlich keines sein sollte, als umgekehrt.

2.4.2 Der Result Pane

Wenn ein Benutzer einen Knoten im *Scope Pane* auswählt, ermöglicht die MMC verschiedene Ansichten des Knotens im *Result Pane*. Diese Ansichten sollen in folgendem Abschnitt erläutert werden. Ähnlich dem Einfügen von Knoten im *Scope Pane* gilt auch in Bezug auf die verschiedenen Ansichten, dass ein *Snap-in* für die Darstellung eines Knoten im *Result Pane* zuständig ist und nicht die MMC selbst.

Jeder Knoten, der in den *Scope Pane* eingefügt wurde, kann auf eine (oder mehrere) der folgenden Arten im *Result Pane* angezeigt werden:

- ***Listen-Ansicht:***
  Standard Darstellungsart im *Result Pane* ist eine Auflistung von Objekten. Diese Objekte repräsentieren typischerweise die Söhne des selektierten Knotens. Angezeigt wird die Bezeichnung und das Symbol des Objektes. Im Gegensatz zum *Scope Pane*, in dem Objekte nur einmal eingefügt werden, bietet der *Result Pane* mehr Dynamik, da hier bei jeder Selektion des Vaterknotens eine Aktualisierung der Anzeige bzw. erneutes Einfügen der Söhne veranlasst wird. Zusätzlich bietet die MMC verschiedene Variationen der Listenansicht, wie die Anzeige großer Symbole, kleiner Symbole und Details.
  In der Detail-Ansicht besteht für den Anwender die Möglichkeit, sich verschiedene Attribute eines *Result Pane*-Objektes in entsprechenden Spalten anzeigen

zu lassen. Unter Verwendung eines dafür vorgesehenen Dialoges kann der Anwender selbst bestimmen, welche Attribute in welcher Reihenfolge angezeigt werden sollen.

- ***Taskpad-Ansicht:***
  Ein *Taskpad* ist eine vereinfachte graphische Präsentation der Aufgaben, die mit einem *Snap-in* durchgeführt werden können. Im Regelfall beziehen sich diese Aufgaben auf den momentan selektierten Knoten. Sinnvoll eingesetzt werden *Taskpads* für Aufgaben wie z.B. Anzeigen von Eigenschaftenfenstern, Ausführen von Menü- bzw. Kommandozeilenbefehlen oder das Öffnen von Web-Seiten.

- ***OCX-Ansicht:***
  In den *Result Pane* können vom *Snap-in* benutzerdefinierte *OLE*-Steuerelemente (*OCX Controls*) eingefügt werden, die auch als ActiveX-Steuerelemente bezeichnet werden.

- ***Web-Ansicht:***
  Der *Result Pane* einer Konsole kann auch als eine Art Web-Browser eingesetzt werden. Wobei sowohl lokale, vom *Snap-in* zur Verfügung gestellte HTML-Seiten, als auch Seiten, die auf externen Web-Servern liegen, angezeigt werden können.

Sofern möglich, soll ein *Snap-in* standardmäßig immer eine Listen-Ansicht zur Verfügung stellen. *Taskpads* sollten nur in Ergänzung zu den verschiedenen Listendarstellungen eingesetzt werden. Auf OCX-Steuerelemente und HTML-Seiten sollte nur dann zurückgegriffen werden, wenn eine Listendarstellung nicht genügend Flexibilität in Bezug auf die Benutzerschnittstelle bietet.
Im *SAT2 Snap-in* wird (standardmäßig) ausschließlich eine Listen-Ansicht eingesetzt. Dabei hat der Anwender die Möglichkeit zwischen den verschiedenen Listen-Varianten zu wählen und bei Verwendung der Detail-Liste natürlich die anzuzeigenden Spalten (Attribute des Objektes) selbst zu bestimmen. Darüber hinaus kann er sich nach belieben eigene *Console Taskpads* zusammenstellen.

2.5 MMC GUI Elemente

Die *Microsoft Management Console (MMC)* ist eine Standard Windows-Applikation. Aus diesem Grund kommen in der graphischen Benutzeroberfläche (*Graphical User Interface*) der MMC und ihrer *Snap-ins* GUI Elemente zum Einsatz, die dem Windows-Standard folgen. Diese GUI Elemente werden auch in vielen anderen Windows-Applikationen verwendet und werden daher als allgemein bekannt vorausgesetzt; Nur der Vollständigkeit halber sollen die Wichtigsten im folgenden Abschnitt kurz beschrieben werden.

Allgemeine GUI Elemente, die in der MMC bzw. in *Snap-ins* zum Einsatz kommen (können), sind:

- ***Symbolleisten***
- ***Menüs***
- ***Dialoge***
- ***Eigenschaftenfenster***
- ***Assistenten***

Eine Besonderheit in der graphischen Benutzeroberfläche der MMC stellen die *Taskpads* dar. Sie zählen zu den MMC spezifischen GUI Elementen und ermöglichen eine graphische Darstellung der Aufgaben, die auf einem selektierten Objekt ausgeführt werden können (siehe Kapitel 2.5.1).
Genau genommen zählt der *MMC Namespace* auch zu den MMC spezifischen GUI Elementen, da er den Kontext für alle MMC Aktivitäten liefert. Dieser wurde aber bereits in Kapitel 2.4 ausführlich behandelt und deshalb an dieser Stelle vernachlässigt.

HINWEIS: Alle im folgenden Abschnitt beschriebenen allgemeinen GUI Elemente, die in *Snap-ins* zum Einsatz kommen, folgen den Standard Windows Konventionen. Deshalb soll an dieser Stelle auf [Msc2] verwiesen werden, wo die festgelegten Konventionen nachgelesen werden können.

2.5.1 Taskpads

Auf Grund der objektorientierten Natur des *Console Tree* tendieren Anwender eher zur Suche nach einem Objekt, das sie bearbeiten wollen, als zur Suche nach einer bestimmten Aufgabe, die erledigt werden soll. Unerfahrenen Anwendern soll diese aufgabenorientierte Darstellung die administrativen Tätigkeiten erleichtern. Andererseits kann in gewissen Situationen eine Kombination von bestimmten Teilen verschiedener *Snap-ins* von Vorteil sein, um eine Aufgabe effizient zu erledigen. In jedem Fall kann ein *MMC Taskpad* erstellt werden, das die Durchführung der Aufgabe vereinfacht.

Taskpads können zwei Kategorien zugeordnet werden:

- ***Console Taskpads***
- ***Snap-in Taskpads***

Die in Version 1.1 der MMC vorgestellten *Snap-in Taskpads* wurden größtenteils durch die *Console Taskpads* der *MMC 1.2* abgelöst. Sie bieten eine Menge an Vorteilen, wie z.B. einfacheres Erstellen, Performance, Konsistenz der Anzeige und große Flexibilität in Bezug auf Benutzerdefinierbarkeit. Deshalb gibt es nur mehr wenige Gründe, den *Snap-in Taskpads* den Vorzug zu geben.

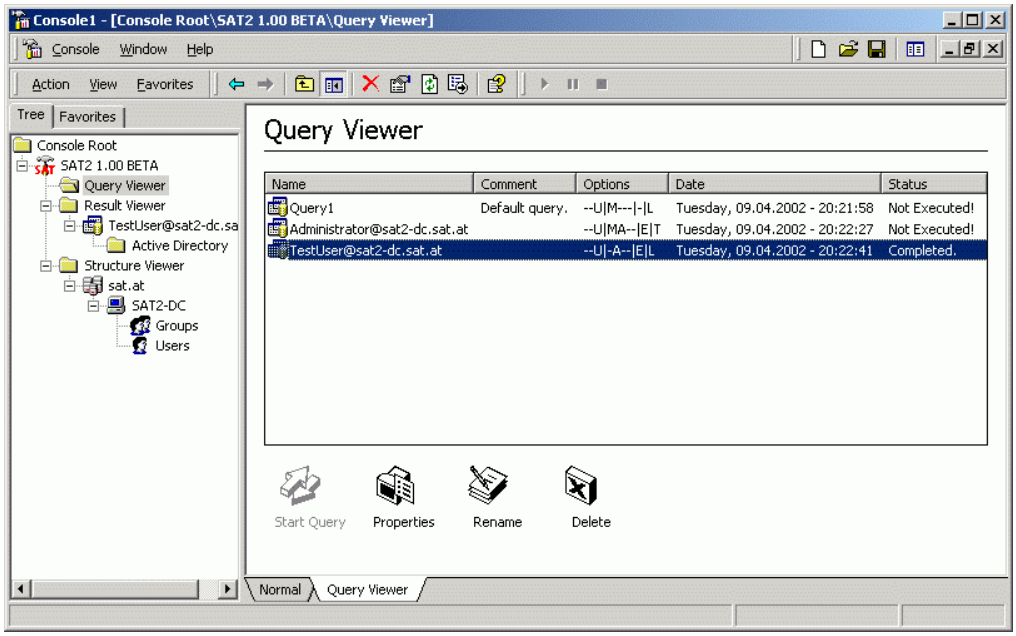


Abbildung 2.6: SAT2 – Query Viewer (Console) Taskpad

Abbildung 2.6 zeigt ein mögliches Konsolen *Taskpad* für den *Query Viewer* des *SAT2 MMC Snap-ins*. Sollten für einen Knoten mehrere *Taskpads* erzeugt werden, sind diese über die zugehörigen Tabs am unteren Ende des *Result Pane* zugänglich.

2.5.1.1 MMC Console Taskpads

Seit dem Erscheinen der *MMC 1.2* kommen hauptsächlich *Console Taskpads* zum Einsatz. Der Anwender kann diese Art der *Taskpads* verwenden um z.B. Assistenten (*Wizards*) zu starten, Eigenschaftenfenster zu öffnen, Menü- oder Kommandozeilenbefehle auszuführen, Web-Seiten zu öffnen, usw.

Console Taskpads sind MMC-seitig implementiert, d.h. die MMC selbst ist für das Erzeugen eines *Taskpads* für einen bestimmten Knoten verantwortlich. Beim Erzeugen eines *Console Taskpads* wird der Anwender durch den *Taskpad Wizard* unterstützt. Auf diesem Weg wird dem Anwender das Eingeben der Informationen, welche die MMC zum Fertigstellen eines *Taskpads* benötigt, erleichtert. Diese Informationen umfassen neben den verschiedenen Darstellungsarten u.a. auch die Menü- und Kommandozeilenbefehle, die mit oder auf dem Objekt ausgeführt werden können.

Die Menübefehle, die vom Anwender ausgewählt werden können, sind äquivalent zu denen, die auch über das Kontextmenü eines selektierten Knotens ausgeführt werden können. Die MMC empfängt die Menübefehle auf die selbe Weise vom *Snap-in*, wie die Kontextmenübefehle. D.h. zwischen den Befehlen eines Kontextmenüs und denen eines MMC-Menüs bzw. in weiterer Folge den Aufgaben eines *Taskpads*, besteht eine 1:1 Beziehung.

Wenn also der Anwender für eine spezielle Aufgabe einen Menübefehl ausgewählt hat und diesen über das *Taskpad* ausführt, kann die MMC die Benutzereingabe an das, für die Ausführung des Befehls zuständige *Snap-in* weiterleiten. Daraus ergibt sich folgender Vorteil: die MMC ist zwar für das Erstellen der *Taskpads* nicht aber für das Durchführen der administrativen Aufgaben zuständig. So kommt die MMC weiterhin „nur“ als Host für Administrations-Tools zum Einsatz und erhält ihre Funktionalität durch die geladenen *Snap-ins*.

2.5.1.2 MMC Snap-in Taskpads

Ein *Snap-in Taskpad* stellt eine spezielle Ansicht eines selektierten Knotens im *Scope Pane* dar. Diese Sicht wird dem Anwender als HTML Seite im *Result Pane* angezeigt und enthält eine Auflistung aller möglichen Vorgänge, die auf dem Knoten ausgeführt werden können. Jeder dieser Vorgänge repräsentiert eine bestimmte Aufgabe und setzt sich aus einem Symbol, einer Bezeichnung, einer Beschreibung und natürlich einem Befehlsmechanismus zur Ausführung der Aufgabe zusammen.

Im Gegensatz zu den *Console Taskpads* hat die MMC hier keinen Einfluss auf das Erzeugen oder die Gestaltung eines *Taskpads*. Dafür ist in diesem Fall auch das *Snap-in* zuständig.

Dem Anwender können drei verschiedene Typen von *Snap-in Taskpads* präsentiert werden:

- ***Standard***
  Im Standard-*Taskpad* wird lediglich dessen Titel und die durchführbaren Aufgaben angezeigt. Dabei kann allerdings auf eine HTML Seite, die von der MMC zur Verfügung gestellt wird, zurückgegriffen werden.

- ***Standard Listenansicht***
  Zusätzlich zum Titel und den, auf einem selektierten Knoten ausführbaren Aktionen, wird ein Listenfeld mit allen zur Auswahl stehenden Objekten (Knoten) angezeigt. Wiederum stellt die MMC die Basiselemente für diese Ansicht zur Verfügung.

- ***Benutzerdefiniert***
  Das *Snap-in* stellt eine eigene HTML Seite mit den möglichen Aufgaben aber gegebenenfalls auch anderen Elementen, zur Verfügung.

Falls ein bestimmter Knoten über eine *Taskpad*-Ansicht verfügt, ist in diesem Fall das *Snap-in* verantwortlich, dem Anwender einen Mechanismus anzubieten, um auf das *Taskpad* zugreifen zu können. Eine Möglichkeit wäre, das *Taskpad* über das Kontext-

menü eines Knotens zugänglich zu machen. Eine andere Lösung hingegen könnte darin bestehen, das *Taskpad* automatisch bei der Auswahl des Knotens anzuzeigen.

Das letztendliche Ausführen einer Aufgabe funktioniert jedoch wieder nach dem bereits bekannten Delegationsprinzip der MMC. Die MMC nimmt die Benutzereingabe(n) entgegen und leitet diese zur Durchführung an das zuständige *Snap-in* weiter.

Obwohl die in diesem Punkt beschriebenen *Snap-in Taskpads* prinzipiell von Entwicklern eingesetzt werden können, wird in den MMC Designrichtlinien davon abgeraten und auf die weitaus effizienteren *Console Taskpads* verwiesen. Aus diesem Grund wurden im *SAT2 Snap-in* keine *Snap-in Taskpads* implementiert, womit der Einsatz von *Console Taskpads* forciert wird.

2.5.2 Symbolleisten (Toolbars)

Symbolleisten bilden die Schnittstelle zu Operationen und Befehlen, die auf Objekten im *Namespace* ausgeführt werden können. Da die Symbolleisten an der Oberseite der Konsolen angeordnet sind, werden sie vom Anwender leicht wahrgenommen und stellen so eine effektive Methode dar, dem Anwender den Befehlssatz eines Tools zugänglich zu machen.

Eine *MMC Konsole* verfügt über zwei verschieden Symbolleistenbereiche. Der eine Bereich (*Console Toolbar Frame*) enthält die zur MMC gehörigen Symbolleisten, der andere (*Snap-in Toolbar Frame*) setzt sich aus den *Snap-in* spezifischen Symbolleisten zusammen. Jeder dieser Symbolleistenbereiche kann mehrere Symbol- und Menüleisten enthalten.

- ***Console Toolbar Frame***
  Dieser Bereich setzt sich aus der Hauptmenüleiste (*Main Menu Bar*) und der Autorenmodussymbolleiste (*Author Mode Toolbar*) zusammen. Beide Teile sind MMC spezifisch und können nicht durch *Snap-ins* verändert oder erweitert werden

- ***Snap-in Toolbar Frame***

  Dieser Bereich besteht aus den Standardmenüs (*Action Menu Bar*), der Standardsymbolleiste (*Common Commands Toolbar*) für Befehle, wie Umbenennen, Löschen oder Aktualisieren eines Objektes und einer bzw. mehreren *Snap-in* spezifischen Symbolleiste(n). Erstere kann, sollte aber aus Flexibilitätsgründen nicht durch das *Snap-in* erweitert werden. Die beiden anderen Symbolleisten sind in Gestaltung und Funktionsumfang von der im *Snap-in* implementierten Funktionalität abhängig.

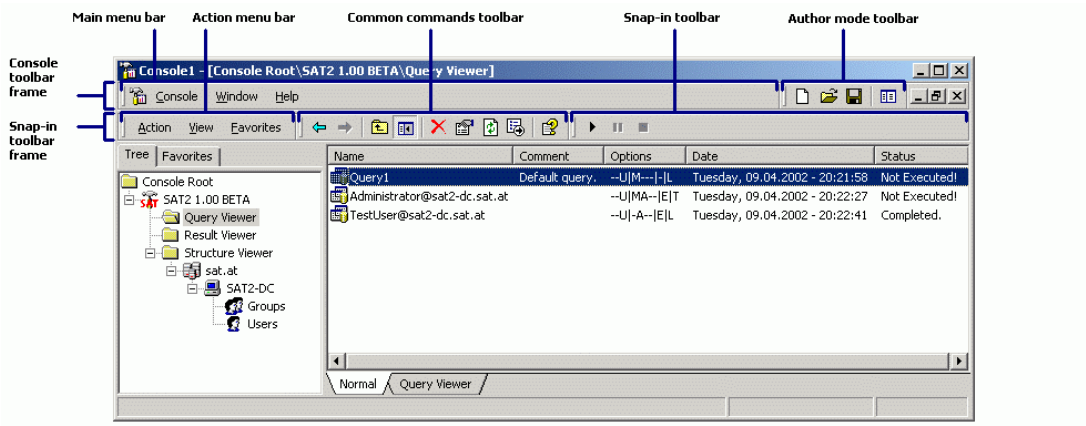


Abbildung 2.7: Symbolleisten

In Abbildung 2.7 sind die verschiedenen Symbolleisten und deren spezielle Bezeichnungen ersichtlich. Als Beispiel hierfür dient der *Query Viewer* des *SAT2 Snap-ins*.

Wenn ein Anwender einen Knoten selektiert, werden zuerst vom – für den selektierten Knoten zuständigen – *Snap-in* alle Symbolleisten, die für diesen Knoten verfügbar sind, in den *Snap-in* Toolbar Frame eingefügt. Anschließend können eventuell vorhandene Extension *Snap-ins* ihre Symbolleisten einfügen. Schaltflächen für Standardbefehle werden, sofern diese Befehle im *Snap-in* implementiert wurden, von der MMC automatisch in der Standardsymbolleiste angezeigt. Empfängt die MMC eine Benutzereingabe über die Symbolleisten, wird die Eingabe an das zuständige *Snap-in* weitergeleitet.

Der Inhalt der Symbolleisten kann sich somit in Abhängigkeit vom Konsolenmodus oder den benutzerdefinierten Einstellungen ändern. Beim Erstellen einer benutzerdefinierten Konsole hat der Anwender die Möglichkeit, die Standardmenüs (Vorgang und

Ansicht), die Standardsymbolleiste und die *Snap-in* Symbolleisten ein- bzw. auszublenden.

2.5.3 Menüs

Die Interaktion mit der *Microsoft Management Console (MMC)* erfolgt hauptsächlich über Menüs. Dabei werden dem Anwender verschiedene Möglichkeiten geboten. Einerseits können Menüs aus der Standardmenüleiste mit den Menübezeichnungen Vorgang (*Action*) und Ansicht (*View*) aufgerufen werden, andererseits werden die mit einem Objekt im *Namespace* assoziierten Befehle durch einen Rechts-Klick auf das Objekt über die sog. Kontextmenüs zugänglich.

Eine Besonderheit im Zusammenhang mit MMC Menüs stellt die Äquivalenz der Standardmenüs (Vorgang und Ansicht) und der Kontextmenüs dar.
Wird vom Anwender ein bestimmter Knoten selektiert, erstellt das *Snap-in* das zugehörige Kontextmenü. Aus den Menüelementen des Kontextmenüs erstellt die MMC anschließend die Standardmenüleiste. So findet der Anwender alle Menüelemente, die im Kontextmenü erscheinen, auch im *Action*-Menü. Der einzige Unterschied in der Menüführung bezieht sich auf das Menü Ansicht (*View*). Dieses ist im Kontextmenü als Menüelement enthalten, wird jedoch beim Erstellen der Standardmenüleiste von der MMC extrahiert und als eigenes Menü angezeigt. Abgesehen davon jedoch sind die beiden Menüs identisch.

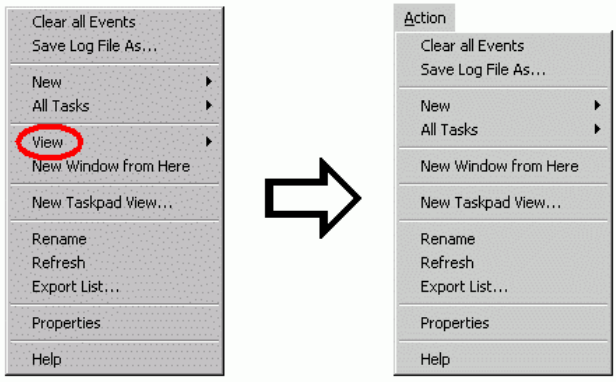


Abbildung 2.8: Kontextmenü (links) und Action-Menü (rechts)

Kontextmenüs werden erstellt, nachdem der Anwender entweder einen Knoten in der Konsolenstruktur oder ein Objekt im *Result Pane* selektiert hat. Dazu liefert die MMC das Grundgerüst und fügt zugleich die Menüelemente für Standardbefehle (z.B. Löschen, Kopieren oder Einfügen), sofern deren Funktionalität im *Snap-in* implementiert wurde, in das Kontextmenü ein. Daraufhin werden alle mit dem Objekt assoziierten Befehle vom *Snap-in* und zuletzt die Befehle der *Extension Snap-ins* hinzugefügt. Für das Anzeigen des Kontextmenüs ist wiederum die MMC zuständig. Sie nimmt auch die Auswahl eines Menüelementes durch den Anwender entgegen und leitet diese an das zuständige *Snap-in* weiter. Das *Snap-in* ist dann für die korrekte Ausführung des Befehls selbst verantwortlich.

Da sich die MMC beim Erstellen des *Action*-Menüs am Kontextmenü eines Knoten orientiert, repräsentieren beide Menüs den Befehlssatz des geladenen *Snap-ins* für den selektierten Knoten.
Dieses Konzept bringt mehrere Vorteile mit sich. So muss das *Snap-in* nicht zwischen gleichen Befehlen, die in verschiedenen Menüs aufgerufen werden, unterscheiden. In Hinblick auf die Implementierung bedeutet das, dass die Funktionalität hinter den Befehlen nicht doppelt – und damit redundant – implementiert werden muss. Da die Menüführung nicht an die MMC gebunden ist, sondern durch das *Snap-in* bestimmt wird und jeder Knoten seine eigenen Menüs haben kann, erhöht sich die Dynamik des *Snap-in*.
Zusätzlich erreicht man eine Steigerung der Benutzerfreundlichkeit, da eine einfache Methode gefunden wurde, Anwendern, die weniger vertraut im Umgang mit Kontextmenüs sind oder *Drop-Down* Menüs bevorzugen, beide Alternativen anzubieten und ihnen keine bestimmte Arbeitsweise aufzuzwingen.

2.5.4 Dialoge

Dialoge (*Dialog Boxes*) werden in der MMC – wie auch in anderen Windows Applikationen – eingesetzt, um Eingaben eines Anwenders, z.B. zum Vervollständigen einer speziellen Aufgabe, entgegenzunehmen. Allerdings sollte darauf geachtet werden, dass die Verwendung von Dialogen auf ein Minimum beschränkt wird. Sie sollten lediglich dann zum Einsatz kommen, wenn zusätzliche Informationen durch den Anwender be-

reitgestellt werden müssen, um einen geplanten *Task* fertig stellen zu können und nicht etwa um gewisse Eigenschaften eines Objektes zu spezifizieren.

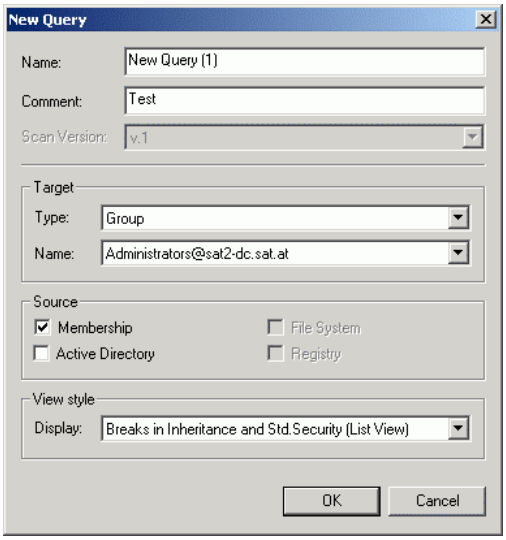


Abbildung 2.9: Dialogfeld

Abbildung 2.9 zeigt den *New Query Dialog* des *SAT2 Snap-ins*. Unter Verwendung dieses Dialoges kann der Anwender neue Abfragen erstellen, welche in weiterer Folge als Basis für die Auswertung der Benutzerberechtigungen zum Einsatz kommen.

Besondere Bedeutung beim Einsatz von Dialogen in der MMC kommt dem Dialogtyp zu. Unter keinen Umständen sollten modale Dialoge in *Snap-ins* verwendet werden. Stattdessen wird empfohlen, nicht-modale Dialoge einzusetzen. Der Grund dafür ist in deren Verhalten zu suchen.
Ein modaler Dialog behält so lange den Fokus, bis er explizit geschlossen wird. In einem *Snap-in* kann diese Eigenschaft äußerst negative Auswirkungen mit sich bringen, die unter Umständen sogar in einem Fehlverhalten des *Snap-in* resultieren kann.

Das Problem lässt sich folgendermaßen schildern. Alle *Snap-ins*, die in der MMC geladen sind, laufen im selben *UI Thread*. Wird von einem *Snap-in* zu einem beliebigen Zeitpunkt ein modaler Dialog aufgerufen, blockiert dieser den Nachrichtenaustausch innerhalb der MMC und somit auch das Empfangen von Nachrichten anderer, geladener *Snap-ins*.

Dazu sei ein keineswegs unrealistisches Szenario, das sich durchaus in der realen Welt ereignen könnte, als Beispiel angeführt:
Man stelle sich vor, in der Verwaltungszentrale der Universität wird u.a. der Parkscheinautomat und die Liftanlage mittels *Snap-ins* überwacht. Das Wechselgeld im Parkautomaten gehe zu Ende, woraufhin das *Snap-in* einen modalen Dialog öffnet, um den Administrator zu informieren, das Wechselgeld nachzufüllen. Im selben Zeitraum tritt ein Alarm der Liftanlage auf. Das dafür zuständige *Snap-in* will ebenfalls eine Benachrichtigung an den Administrator senden, ist dazu aber nicht in der Lage, da der bereits geöffnete Wechselgeld-Dialog alle Nachrichten blockiert. Für Personen, welche sich möglicherweise im Aufzug befinden, könnte dies unter Umständen verheerende Auswirkungen haben.

Dialoge werden in der MMC aber nicht nur eingesetzt, um Benutzereingaben entgegenzunehmen, sondern auch um den Anwender über die Konsequenzen einer unwiderruflichen Aktion aufmerksam zu machen oder um Fehlermeldungen anzuzeigen.

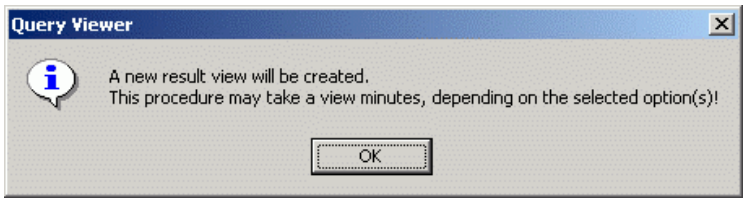


Abbildung 2.10: Nachrichtenfenster

Abbildung 2.10 zeigt beispielsweise eines der sogenannten Nachrichtenfenster (*Message Box*). Dieses Nachrichtenfenster wird im *SAT2 Snap-in* dazu verwendet, den Anwender über den Start einer Analyse der Zugriffsrechte zu informieren.

Allerdings stehen diese Nachrichtenfenster in Widerspruch zur o.a. Problematik. Windows Nachrichtenfenster fallen in die Kategorie der modalen Dialoge und sollten deshalb möglichst selten verwendet werden. Außerdem wirken sie oft störend oder tragen mehr zur Verwirrung des Anwenders als zu dessen Unterstützung bei. In gewissen Situationen können diese Nachrichtenfenster jedoch nützlich sein. Es liegt also am *Snap-in* Entwickler, sich für den richtigen Einsatz des passenden Instruments zur Darstellung von Informationen, Warnungen oder Fehler, zu entscheiden.

Als Alternative bietet die MMC die sogenannten *Snap-in Message Boxes* an (siehe Abbildung 2.11). Dabei werden Nachrichten im *Result Pane* des selektierten Knotens angezeigt.

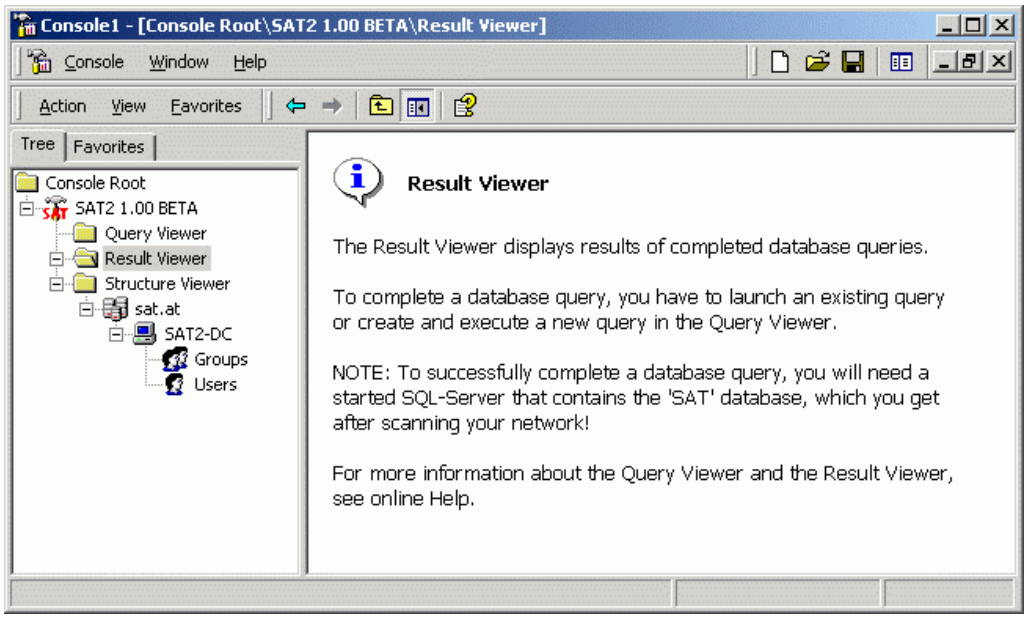


Abbildung 2.11: Snap-in Message Box

Durch das großzügige Platzangebot im *Result Pane* können Warnungen oder aufgetretene Fehler detailliert beschrieben und Lösungsvorschläge in übersichtlicher Form präsentiert werden. Weiters können Nachrichten in den *Snap-in Message Boxes* über einen beliebig langen Zeitraum, z.B. bis ein Fehler behoben wurde, angezeigt werden und verschwinden nicht, wie herkömmliche Windows Nachrichtenfenster, nachdem sie geschlossen wurden. Wesentlich ist auch, dass ein Anwender auf diese Art leicht feststellen kann, welches Objekt eine Nachricht anzeigt oder einen Fehler ausgelöst hat.

2.5.5 Eigenschaftenfenster (Property Sheets)

Einen wesentlichen Bestandteil der MMC in Bezug auf die Benutzerinteraktion stellen die Eigenschaftenfenster dar. Anwender können auf diesem Weg Attribute oder Einstellungen eines Objektes betrachten und gegebenenfalls verändern. MMC Eigenschaftenfenster basieren auf dem Konzept erweiterbarer, nicht-modaler Dialoge.

Dadurch kann der Anwender, trotz eines geöffneten Eigenschaftenfensters, auch auf andere (Eigenschaften-)Fenster zugreifen. Das Prinzip der Erweiterbarkeit ist deshalb so

wichtig, da *Snap-ins* die Möglichkeit haben müssen, ihre eigenen Eigenschaftenseiten darstellen zu können.
Obwohl prinzipiell die Möglichkeit besteht, die Eigenschaften eines Objektes im *Result Pane* des *Snap-in* anzuzeigen, sollte der Anwender durch die Eigenschaftenfenster in der Lage sein, alle Eigenschaften eines bestimmten Objektes von einem zentralen Ort aus betrachten und eventuell auch verändern zu können.

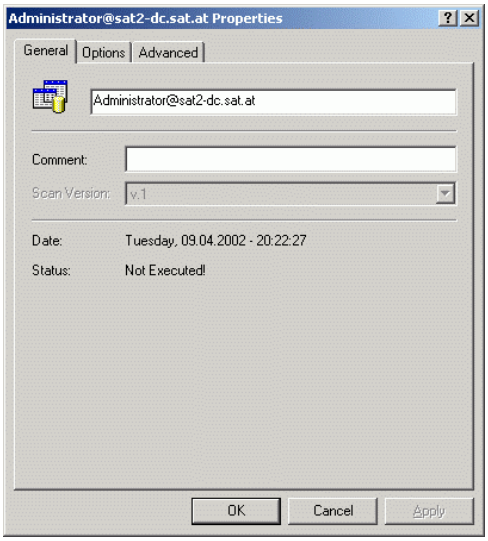


Abbildung 2.12: Eigenschaftenfenster

Abbildung 2.12 zeigt das Eigenschaftenfenster einer Abfrage, die im *SAT2 Snap-in* erstellt wurde und zugehörige Eigenschaftenseiten, in welchen sowohl statische als auch editierbare Eigenschaften des selektierten Objekts angezeigt werden (können).

Das Eigenschaftenfenster eines Objektes kann entweder über die Eigenschaftenschaltfläche in der Standardsymbolleiste oder über den Befehl „Eigenschaften“ im Kontextmenü, das angezeigt wird, wenn der Anwender mit der rechten Maustaste auf ein Objekt im *Scope Pane* bzw. *Result Pane* klickt, aufgerufen werden.

Wird also ein Eigenschaftenfenster eines Objektes aufgerufen, so ist die MMC für dessen Darstellung verantwortlich. *Snap-ins* können dann ihre eigenen Eigenschaftenseiten hinzufügen. Jede Eingabe eines Anwenders, die die MMC empfängt, wird an das *Snap-in* weitergeleitet, das die Eigenschaftenseite eingefügt hat.

Eigenschaftenseiten sollten auch dazu genutzt werden, die Eigenschaften eines Objektes in verschiedene Kategorien zu unterteilen und zu gruppieren. Dies wird durch die sogenannten Registerkarten unterstützt. Jeder Tab in dieser Registerkarte repräsentiert eine bestimmte Eigenschaftenseite, die nach einem Mausklick des Anwenders auf den zugehörigen Tab angezeigt wird. Dadurch erhält der Anwender die Möglichkeit problemlos zwischen den einzelnen Seiten hin und her zu blättern.
Die Anordnung der Eigenschaften bzw. Eigenschaftenseiten soll aus Gründen der Benutzerfreundlichkeit, beginnend bei den am häufigsten bis hin zu den am wenigsten Gebräuchlichen, erfolgen. Zusätzlich empfiehlt sich in gewissen Situationen der Einsatz von Sekundärfenstern, zur Anzeige erweiterter Eigenschaften bzw. selten bis nie benötigter Attribute oder Einstellungen eines Objektes.

2.5.6 Assistenten (Wizards)

In einem *MMC Snap-in* können Assistenten entweder zum Durchführen komplexer Aufgaben oder zur Unterstützung unerfahrener Anwender eingesetzt werden. Ein Assistent kann zu jedem beliebigen Zeitpunkt aufgerufen werden, beispielsweise beim Hinzufügen eines *Snap-ins* zu einer Konsole oder als Antwort auf ein Menü- oder Symbolleistenkommando.

Ein Assistent automatisiert die Durchführung einer Aufgabe dahingehend, dass der Anwender Schritt für Schritt seinem Ziel näher gebracht wird. Seit der Einführung dieser *Wizards* in Windows Umgebungen, hat sich deren Einsatz, teilweise mit großem Erfolg, kontinuierlich verbreitet. Somit finden sie auch in der MMC und deren *Snap-ins* ihre Verwendung.

Abbildung 2.13 zeigt die erste Seite des *New Taskpad View Wizards*, der beispielsweise im *SAT2 Snap-in* verwendet werden kann, um eines, der in Kapitel 2.5.1 beschriebenen *Console Taskpads*, zu erstellen. Wie alle Assistenten, welche in der MMC zum Einsatz kommen, unterliegt auch dieser dem *Wizard97*-Stil für Windows.

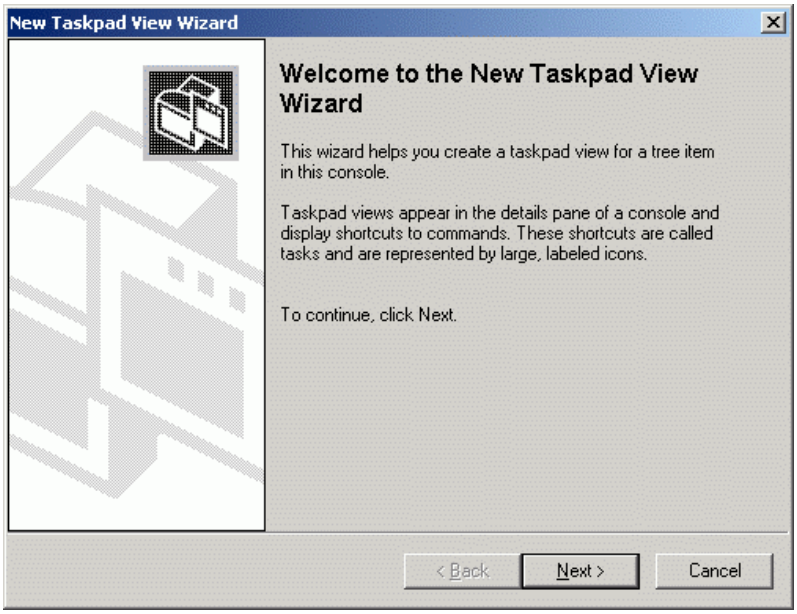


Abbildung 2.13: New Taskpad View Wizard – Startseite

Obwohl Assistenten hervorragend geeignet sind, um unerfahrene Anwender zu unterstützen, sollten sie nicht die einzige Möglichkeit zur Durchführung einer Aufgabe sein. Erfahrene Anwender fühlen sich unter Umständen durch den Assistenten in ihrer Arbeitsweise eingeschränkt.

2.6 Systemvoraussetzungen

Seit der Entwicklung des Prototyps der *Microsoft Management Console (MMC)* im Jahre 1996 hat sich am Betriebssystemsektor vieles verändert. In dieser Zeit wurde hauptsächlich das Windows NT Betriebssystem als Basis für Microsoft Netzwerke eingesetzt.

Das wesentlich jüngere Windows 2000 oder das kürzlich erschienene Windows XP Betriebssystem sollen das weit verbreitete, aber teilweise „veraltete“ Windows NT vor allem im Bereich der Netzwerkorganisation aber auch im Heimanwenderbereich ablösen. Deshalb verfolgt Microsoft auch die Strategie, die MMC auf möglichst allen Windows-Plattformen für Administrationsaufgaben einzusetzen.

Die MMC wurde in der Windows NT Ära entwickelt und war damals eigentlich für den Einsatz auf dem Gebiet der Netzwerk-Administration vorgesehen. Die Vielseitigkeit

und Flexibilität, welche mit der MMC u.a. durch das *Snap-in* Konzept erreicht werden kann, führte aber bald dazu, dass Administrations-Tools auf Basis der *MMC Konsolen* auch zur lokalen Computerverwaltung eingesetzt wurden. Vor allem seit dem Erscheinen der Windows 2000 Systeme kommen diese Tools vermehrt zum Einsatz und sollen künftig die Durchführung nahezu aller Administrationsaufgaben ermöglichen.

Die (aktuellen) Versionen 1.1 und 1.2 der MMC laufen unter folgenden Betriebssystemen:

- ***Windows NT 4.0 (incl. Service Pack 3)***
- ***Windows 2000 (ab Beta 3)***
- ***Windows 95/98/Me***

MMC 2.0 wurde ausschließlich für Windows XP entwickelt. Deshalb wird deren Einsatz in Windows NT 4.x bzw. Windows 2000 Systemen seitens Microsoft nicht unterstützt. Dies bedeutet aber keinesfalls, dass diese Version der MMC in den o.a. Betriebssystemen nicht lauffähig ist, sondern lediglich, dass Microsoft einen korrekten Betrieb nicht garantiert und durch den Versionskonflikt hervorgerufene Fehler nicht oder nur in Bezug auf Windows XP korrigieren wird. Darüber hinaus wird von *Microsoft Support* zu dieser Thematik auch keine Unterstützung bzw. Hilfestellung angeboten.

Softwareentwickler haben die Möglichkeit, *MMC Snap-ins* (für alle o.a. Versionen der MMC) in verschiedenen Entwicklungsumgebungen bzw. Programmiersprachen zu implementieren. Dabei stehen folgende Sprachen zur Auswahl:

- ***Microsoft Visual C++ 5.0 oder 6.0***
- ***Microsoft Visual Basic 6.0***

Zusätzlich soll die Entwicklung der *Snap-ins* durch spezielle *Wizards*, die für beide Sprachen existieren, vereinfacht werden, obgleich deren Verwendung nicht empfohlen wird. Denn sowohl der *ATL/COM Application Wizard* für *Visual C*, als auch der *Snap-*

in Designer für *Visual Basic* sollen nach [Msng] künftig von Microsoft weder weiterentwickelt noch unterstützt werden.

Der in [Mmc1] bzw. [Msng] empfohlene Weg, eigene *Snap-ins* zu entwickeln, wird immer wieder deutlich gemacht. Dabei soll auf Basis der *Platform SDK* Beispiele ohne Verwendung des *ATL Wizards* in *Microsoft Visual C++* entwickelt werden.

Letztendlich bleibt die Entscheidung für diese oder jene Entwicklungsumgebung dem Entwickler selbst überlassen. Die Unterstützung seitens *Microsoft Support*, die vorhandenen *Code*-Beispiele und die höhere Performance, die mit *Visual C/C++* erreicht werden kann, sollten aber dennoch als Entscheidungsgrundlage nicht außer Acht gelassen werden.

MMC und .NET?

Diese Frage ist auch im Zusammenhang mit dem *Projekt SAT2*, auf der Suche nach „zeitgemäßen" Technologien für die Visualisierungs-Applikation, des öfteren ans Tageslicht getreten.

Bis dato wurde keine (offizielle) Stellungnahme seitens Microsoft zu dieser Thematik gefunden. Einzig in [Msng] wurde der Einsatz der MMC im *.NET Framework* und die Implementierung der *Snap-ins* in *C#* diskutiert.

Um die Antwort vorweg zu nehmen, die Kombination MMC und *.NET* ergibt – jedenfalls derzeit – keinen Sinn und ist deshalb nicht empfehlenswert. Auch wenn prinzipiell die Möglichkeit besteht, *Snap-ins* in *C#* oder anderen *.NET* Technologien zu entwickeln – immerhin müssen „nur" die gleichen Interfaces in einer anderen Programmiersprache implementiert werden – sollte alleine auf Grund der zu erwartenden Performance-Einbußen davon abgelassen werden.

Da weder die MMC, noch ihre *Snap-ins* in *Marshalled Code* laufen, dies aber die Grundvoraussetzung für „echte" *.NET* Applikationen darstellt, muss in jedem Fall mit starken Einbußen in Bezug auf die Performance gerechnet werden. Abgesehen davon

könnten *Snap-ins*, die beispielsweise in *C#* entwickelt wurden, nur auf *.NET* Plattformen eingesetzt werden.

Aus diesen Gründen wurde C# nicht in das MMC-Testprogramm aufgenommen. Das hat somit zur Folge, dass beim Auftreten von Fehlfunktionen nicht auf eine Unterstützung durch *Microsoft Support* gehofft werden kann, aber auch, dass trotz möglicherweise bekannter Fehler, dahingehend keine Korrekturen vorgenommen werden (können). Mit einer Erweiterung oder Adaption der MMC ist in absehbarer Zeit nicht zu rechnen, gegebenenfalls wird sie durch andere Technologien ersetzt werden.

2.7 Implementierungstechnische Grundlagen

Der folgende Abschnitt soll dem Leser einen kurzen Einblick in die Implementierung eines *MMC Snap-ins* gewähren. Wesentlich ist, dass dieser Teil keine eigenständige Anleitung zur Erstellung eines *Snap-ins* darstellt, sondern einen Überblick bieten soll, was alles zu berücksichtigen und erledigen ist, um letztendlich ein funktionstüchtiges *Framework* zu erhalten. Dieses Grundgerüst stellt jedoch lediglich die Basis eines *Snap-ins* dar und enthält soweit noch keine administrative Funktionalität. Diese muss in Abhängigkeit von den Anforderungen, denen das *Snap-in* entsprechen soll, zusätzlich implementiert werden.

Die Entwicklung eines *Snap-ins* ist alles andere als trivial. Die Komplexität, welche die MMC in sich birgt, sollte nicht unterschätzt werden. Mit Sicherheit kann behauptet werden, dass ein Durchlesen der Dokumentation der verschiedenen Methoden und Interfaces keineswegs ausreicht, um eigene *Snap-ins* entwickeln zu können. Grundvoraussetzung für die erfolgreiche Entwicklung eines *Snap-ins* ist zweifellos ausreichendes Verständnis für die Funktionsweise der MMC.

Der Grundstein für dieses Verständnis soll im folgenden Abschnitt gelegt werden. Deshalb wurde auch auf die Abbildung von (detaillierten) *Sourcen* an dieser Stelle verzichtet und die Beschreibung der relevanten Teile des *Snap-in Basis-Frameworks* in den Vordergrund gestellt. Dennoch soll hier auf [Mmc2] verwiesen werden. Dort findet man

genaue Anleitungen und Beispiele zur Implementierung der für ein solches *Framework* benötigten Methoden und Interfaces.

In Kapitel 3 dieser Diplomarbeit wird das *Projekt SAT2* und im Speziellen das *SAT2 MMC Snap-in* beschrieben. In diesem Zusammenhang wird dann u.a. die Implementierung der Analysefunktionen – also die zuvor erwähnte administrative Funktionalität – erläutert (siehe Kapitel 3.7.7)

2.7.1 Das Snap-in Basis-Framework

Wie bereits erwähnt wird die Funktionalität einer *MMC Konsole* durch das *Snap-in* bestimmt. Um diese Funktionalität jedoch nutzen zu können, müssen spezielle Methoden und Interfaces implementiert werden, die eine Kommunikation des *Snap-ins* mit der MMC ermöglichen.

Diese Kommunikation erfolgt über eine *Component Object Model (COM)* Schnittstelle (siehe dazu Abbildung 2.3) und könnte beispielsweise folgendermaßen ablaufen:
Die MMC nimmt eine bestimmte Benutzereingabe entgegen und leitet diese über die COM Schnittstelle an das *Snap-in* weiter. Das *Snap-in* führt die mit der Eingabe assoziierte Aktion aus. Dann benachrichtigt es - wiederum über die COM Schnittstelle - die MMC, wie der Anwender von der erledigten oder u.U. auch fehlgeschlagenen Aufgabe in Kenntnis zu setzten ist. Daraufhin sorgt die MMC für die Darstellung des Resultats der Aktion in der vom *Snap-in* definierten Form.

Dieses Konzept ermöglicht eine gewisse Unabhängigkeit zwischen MMC und *Snap-in*, d.h. die MMC benötigt keinerlei Wissen über ein *Snap-in*, das in einer Konsole geladen wurde, weder über dessen Einsatzgebiet, noch über Technologien oder Mechanismen, die zur Durchführung der administrativen Aufgaben verwendet werden.

Zu diesem Zweck werden *Snap-ins* als sogenannte *COM In-process Server Dynamik Link Libraries (DLLs)* implementiert. Die Bedeutung dieses äußerst komplizierten Begriffs lässt sich aber in wenigen Worten relativ einfach erklären.

Das Komponentenobjektmodell COM ist eine objektorientierte Architektur zur Erzeugung wiederverwendbarer Objekte, die auf dem *Client/Server* Prinzip aufbaut. Dabei übernimmt die MMC die Rolle des *Servers*. Sie stellt spezielle Methoden und Interfaces zur Verfügen, die vom *Client*, der durch das *Snap-in* repräsentiert wird, genutzt werden können. In diesem Fall laufen sowohl *Client* als auch *Server* im selben Prozessraum (*In-process Server*), müssen aber dennoch von verschiedenen Prozessen genutzt werden können. Deshalb ist die Implementierung als DLL naheliegend.

Das Grundgerüst eines jeden *Snap-ins* bilden zwei MMC-spezifische und zwei Standard-COM Interfaces, die auf jeden Fall implementiert werden müssen. Durch die Implementierung dieser Interfaces erhält das *Snap-in* seine Basisfunktionalität und damit die Möglichkeit zur Kommunikation mit der MMC.

- ***Standard COM Interfaces***
  - *IDataObject:* Realisiert den Datenaustausch innerhalb der MMC. Durch die Methoden dieses Interfaces können der MMC Informationen über Objekte im *Namespace* zur Verfügung gestellt werden.
  - *IClassFactory:* Bildet die Basis für die Handhabung von COM-Objekten. Ermöglicht das Anlegen neuer COM-Objekte und das Laden des COM-Servers in den Hauptspeicher.

- ***MMC spezifische COM Interfaces***
  - *IComponentData*: Ermöglicht die Kommunikation des *Snap-ins* mit der MMC und steht in engem Zusammenhang mit den Objekten, die im *Scope Pane* angezeigt werden.
  - *IComponent*: Ist ebenfalls die Kommunikation des *Snap-ins* mit der MMC zuständig. Im Gegensatz zu *IComponentData* ist dieses Interface hauptsächlich für Objekte im *Result Pane* zuständig.

HINWEIS: Die o.a. Interfaces und deren Methoden werden häufig aktiviert, deshalb ist auf eine effiziente (und korrekte) Implementierung besonderer Wert zu legen.

2.7.2 Registrieren des Snap-ins

Da *Snap-ins* als *COM In-process Server DLLs* implementiert werden, müssen sie sich selbst in einem speziell für *Snap-ins* vorgesehenen Bereich in der Windows Registrierungsdatenbank (*Registry*) registrieren. Diesen Bereich, in dem Daten über alle in einem System verfügbaren *Snap-ins* gespeichert sind, findet man unter folgendem Pfad:

- **`HKEY_LOCAL_MACHINE\Software\Microsoft\MMC\SnapIns`**

Durch die in der *Registry* gespeicherten Daten erhält die MMC wesentliche Informationen über das *Snap-in*. Diese Informationen dienen u.a. dazu, das *Snap-in* zu lokalisieren und bei Bedarf in die Konsole zu laden.
Damit *Snap-ins* eindeutig identifiziert und eventuelle Namenskonflikte ausgeschlossen werden können, müssen sie über einen sogenannten *Class Identifier (CLSID)* verfügen.

Um die Eindeutigkeit der *CLSIDs* gewährleisten zu können, müssen diese, unter Verwendung speziell dafür vorgesehener Programme (z.B. guidgen.exe), vom *Snap-in* Entwickler generiert werden.

Anmerkung: *Class Identifier* (*CLSID)* ist ein COM spezifischer Ausdruck und wird in diesem Zusammenhang auch äquivalent zur Bezeichnung *Global Unique Identifier (GUID)* verwendet. Beide Begriffe bezeichnen weltweit eindeutige, alphanumerische Zeichenketten, die zur Identifikation spezieller Objekte eingesetzt werden. Diese CLSIDs bzw. GUIDs erkennt man daran, das sie von geschlungenen Klammern eingeschlossen sind.

Der Schlüssel unter dem ein *Snap-in* in der Registrierungsdatenbank gespeichert wird, lässt sich folgendermaßen darstellen:

- ***Allgemeine Form:***

```
HKEY_LOCAL_MACHINE\Software\Microsoft\MMC\SnapIns\
   {snapinCLSID}
      About REG_SZ "{SnapinAboutCLSID}"
      NameString REG_SZ "SnapinDisplayName"
      StandAlone
      NodeTypes
         {nodetypeGUID}
```

- ***Beispiel – Registry-Key des SAT2 Snap-ins:***

```
HKEY_LOCAL_MACHINE\Software\Microsoft\MMC\SnapIns\
   {EEF0E5B7-D30B-4A75-9CD9-2B6FEFDA1384}
      About REG_SZ "{19FF228E-6D88-4BD7-B201-6060F70F6429}"
      NameString REG_SZ "SAT2 1.00 BETA"
      Provider REG_SZ "SAT-Team,FIM-University of Linz,Austria"
      Version REG_SZ "1.00B"
      StandAlone
```

Erwähnenswert ist, dass die zuvor angeführte allgemeine Form eines *Registry-Keys* erweitert werden bzw. in Abhängigkeit des *Snap-in* Typs variieren kann (siehe *SAT2 Snap-in Registry-Key* Beispiel).

Im folgenden Abschnitt erfolgt eine kurze Erklärung der einzelnen Bezeichnungen bzw. Werte.

- ***{snapinCLSID}***

Bezeichnet jene Zeichenkette, welche den CLSID des *Snap-in* in der Registrierung repräsentiert.

- ***About***

Dieser Wert bezeichnet wiederum die Stringrepräsentation eines CLSID. In diesem Fall identifiziert der CLSID aber das *ISnapInAbout* Objekt (siehe Kapitel 2.7.3), das für die Darstellung von Symbol und Beschreibung des *Snap-in* im „*Snap-in hinzufügen/entfernen*" Dialog verantwortlich ist.

- ***NameString***

Spezifiziert den Namen des *Snap-in*, der im „*Snap-in hinzufügen/entfernen*" Dialog angezeigt wird.

- ***StandAlone***

Dieser Schlüssel kennzeichnet den Typ des *Snap-ins*. *StandAlone Snap-ins* müssen diesen Wert setzen und können dadurch als solche in den *Namespace* eingefügt werden. *Extension Snap-ins* dürfen diesen Schlüssel nicht speichern. Sie müssen sich zusätzlich an einer anderen Stelle der *Registry*, unter dem Schlüssel des *Snap-ins* bzw. Knotens,

der erweitert wird, registrieren. *Dual-mode Snap-ins* müssen sich sowohl als *StandAlone* als auch als *Extension Snap-in* registrieren.

- ***NodeTypes***

Unter diesem Schlüssel werden die *GUIDs* aller Knoten des *Snap-in* als Sub-Schlüssel gespeichert, die von *Extension Snap-ins* erweitert werden können. Anhand der hier gespeicherten Schlüssel können alle für das *Snap-in* zulässigen Erweiterungen in der *Registry* gefunden werden.

- ***{nodetypeGUID}***

Dieser Wert stellt die Stringrepräsentation des *GUID* eines erweiterbaren Knotens dar. Zusätzlich muss dieser *GUID* unter folgendem Pfad in der *Registry* gespeichert werden:

- **`HKEY_LOCAL_MACHINE\Software\Microsoft\MMC\NodeTypes\`*`{nodetypeGUID}`***

Unter diesem Schlüssel werden dann alle *Extension Snap-ins* registriert, die diesen Knoten erweitern.

Um die Registrierung durchführen zu können, müssen vom Entwickler für jedes *Snap-in* folgende *COM API Funktionen* implementiert werden:

- ***DllRegisterServer, DllUnregisterServer:***
  Diese Funktionen übernehmen das Erzeugen, Speichern und Löschen von Registrierungseinträgen für *COM In-process Server*.

- ***RegisterSnapIn, UnregisterSnapIn:***
  Diese Funktionen sind für korrektes Registrieren eines *Snap-ins* unter Berücksichtigung der o.a. Konventionen und Löschen der Einträge aus der *Registry* zuständig.

2.7.3 „About“ – Informationen

In nahezu allen Windows Applikationen wird dem Anwender die Möglichkeit geboten, Informationen über Version und Hersteller der Applikation zu betrachten. Diese Infor-

mationen sind zumeist über ein im Hilfemenü platziertes, spezielles Menüelement mit der Bezeichnung *„Info“* oder *„About“* zugänglich und werden als Dialogfenster angezeigt.

Die *Microsoft Management Console (MMC)* unterstützt diese Art der Informationsvermittlung ebenfalls. Dabei werden die, in der Implementierung eines *Snap-ins*, spezifizierten Daten in einem Eigenschaftenfenster dargestellt. Im Gegensatz zu herkömmlichen Windows Applikationen können diese Informationen aber nicht über ein Menüelement der Menüzeile abgerufen werden. Der Anwender findet diese Informationen ausschließlich im „*Snap-in hinzufügen/entfernen*“ Dialog und erhält erst durch die Selektion eines der in diesem Dialog aufgelisteten *Snap-ins* Zugriff darauf. Außerdem können diese *„About“*-Informationen nur dann aufgerufen werden, wenn die Konsole im *Autorenmodus* betrieben wird.

Um die *„About“*-Informationen anzeigen zu können, muss das *ISnapinAbout Interface* implementiert werden. Durch die Implementierung der Methoden dieses Interfaces können folgenden Informationen spezifiziert werden:

- ***Snap-in spezifisches Symbol (Icon)***
- ***Snap-in Name***
- ***Hersteller***
- ***Versionsnummer***
- ***Beschreibung***

Darüber hinaus kann in der Implementierung dieses Interfaces ein benutzerdefiniertes Symbol für den *Static Node* des *Snap-ins* festgelegt werden, welches von der MMC zur Anzeige des Wurzelknotens verwendet wird. Sollte das *ISnapinAbout Interface* nicht implementiert werden, greift die MMC auf Standardsymbole zurück.

Ein wesentlicher Punkt für die Implementierung dieses Interfaces ist, dass der *CLSID*, der *ISnapinAbout* Objekte eines *Snap-ins* bezeichnet, zum Schlüssel des *Snap-ins* in der Registrierung gespeichert werden muss (siehe Kapitel 2.7.2).

2.7.4 MMC Namespace

Der *MMC Namespace* setzt sich, wie in Kapitel 2.4 bereits beschrieben, aus dem *Scope Pane* und dem *Result Pane* zusammen. Durch die Implementierung bestimmter Klassen und Methoden können Objekte (*Items*) in den *Namespace* eingefügt werden. Somit erhält das *Snap-in* seine eigentliche, administrative Funktionalität. Dabei wird festgelegt, welche Objekte eingefügt werden können und welche Aufgaben auf diesen Objekten ausführbar sind.

Neben der Durchführung spezieller Aufgaben ist eine *Snap-in* auch dafür verantwortlich, der MMC Benachrichtigungen zu senden, wie Objekte dargestellt werden und welches Ereignis auf eine bestimmte Benutzereingabe folgt. Der Grundstein für diese Kommunikation wurde mit dem *Snap-in Basis-Framework* gesetzt (siehe 2.7.1). Die Interfaces, welche zu diesem Zweck implementiert wurden, stellen u.a. Methoden zur Verfügung, mit denen diese Benachrichtigungen verschickt werden können.

Der *Scope Pane* enthält Objekte, die auch als Knoten des *Console Tree* oder im programmiertechnischen Sinn als *Scope Items* bezeichnet werden. Sie repräsentieren in der Regel verschiedene Ordner oder Container. Allen voran findet man den Wurzelknoten eines *Snap-in*, den *Static Node*. Dieser Knoten verfügt normalerweise über keine spezielle Funktionalität, er fungiert lediglich als statischer Vaterknoten für alle anderen Objekte, die in den *Namespace* eingefügt werden. Zur Manipulation der *Scope Items* stellt die MMC folgende Interfaces zur Verfügung, die je nach Bedarf zu implementieren sind.

- ***IComponentData***
- ***IConsoleNamespace2***
- ***IConsole2***

Da die Kommunikation der *Scope Items* über das *IComponentData Interface* erfolgt, sei es in diesem Zusammenhang nochmals angeführt. Die Methoden, die von diesen Interfaces angeboten werden, dienen z.B. zum Einfügen oder Entfernen von *Scope Items* oder zum Expandieren eines Knotens.

Im *Result Pane*, der das Gegenstück zum *Scope Pane* im *MMC Namespace* darstellt, werden dem Anwender mehr oder weniger umfangreiche Informationen über ein selektiertes Objekt im *Scope Pane* angezeigt. Je nach Struktur und Aufbau des *Console Tree* werden im *Result Pane* sowohl *Scope Items* als auch *Result Items* dargestellt. Um diese Objekte entsprechend einfügen, darstellen oder manipulieren zu können, stellt die MMC wiederum eine Vielzahl an Interfaces und Methoden zur Verfügung. Zusätzlich werden Funktionen benötigt, um die in Kapitel 2.4.2 beschriebenen Ansichten realisieren zu können.

Erwähnenswert ist in diesem Zusammenhang die Kommunikation der *Result Items* eines *Snap-ins* mit der MMC. Diese basiert zwar auf dem selben Prinzip, wie die der *Scope Items*, erfolgt jedoch über das Zweite der beiden zuvor erwähnten MMC spezifischen COM Interfaces, das *IComponent Interface*. Da allerdings nicht nur das Kommunikations-Interface ein anderes ist, sondern *Scope Items* und *Result Items* auch unterschiedlich behandelt werden müssen, ist bei der Implementierung unbedingt auf eine exakte Separierung der Objekte zu achten.

Weitere Informationen über die Implementierung des *MMC Namespace* können an dieser Stelle, auf Grund der hohen Komplexität der Thematik nicht geboten werden. Eine detaillierte Beschreibung aller verfügbaren Interfaces und zugehöriger Methoden, die zur Manipulation der Objekte im *MMC Namespace* implementiert werden können und deren Zusammenspiel, würde den Rahmen dieses Kapitels sprengen. Deshalb sei an dieser Stelle nochmals auf [Mmc2] verwiesen.

2.7.5 GUI Elemente

GUI Elemente stellen einen der wichtigsten Bestandteile der MMC in Bezug auf Benutzerinteraktion dar. Deshalb ist in diesem Zusammenhang unbedingt auf eine korrekte Implementierung, vor allem in Hinblick auf die Benutzerfreundlichkeit, besonderer Wert zu legen.

Die Verarbeitung von Benutzereingaben basiert in allen Bereichen der MMC auf dem Delegationsprinzip. Dabei werden von der MMC die Eingaben des Anwenders entge-

gengenommen und an das zuständige *Snap-in* weitergeleitet. Das *Snap-in* wiederum verarbeitet diese Eingabe und teilt der MMC mit, wie diese dem Anwender das Resultat der durchgeführten Aktion darstellen soll.

Da bestimmte GUI Elemente sowohl zur Manipulation von *Scope Items* als auch *Result Items* zum Einsatz kommen, also objektbezogen sind, werden die GUI Interfaces auch in Zusammenhang mit den Objekten (d.h. innerhalb der Klasse von der ein *Scope Item* oder *Result Item* abgeleitet wird) implementiert.

Zusätzlich existieren Interfaces, die MMC-seitig implementiert wurden. Dadurch wird dem Entwickler die Arbeit erleichtert. Er kann diese Funktionalität nutzen, ohne den Programmcode dafür selbst schreiben zu müssen. Diese speziellen Interfaces und Methoden dürfen jedoch unter keinen Umständen überschrieben werden.

Folgenden Tabelle bietet einen Überblick über jene Interfaces, die zur Implementierung der GUI Elemente eingesetzt werden können. Die Dokumentation dieser Interfaces und entsprechende Programmbeispiele zur Implementierung können in [Mmc2] nachgelesen werden.

GUI Element	Interface
Taskpad	IExtendTaskPad
Symbolleisten	IExtendControlbar
Kontextmenüs, Menüs	IExtendContextMenu
Eigenschaftenfenster	IExtendPropertySheet IExtendPropertySheet2
Assistenten	IExtendPropertySheet IExtendPropertySheet2

Tabelle 2.1: MMC GUI Elemente und Interfaces

HINWEIS: Da, wie zuvor erwähnt, auf die Benutzerfreundlichkeit besonderes Augenmerk zu legen ist, wurden von Microsoft speziell für diesen Zweck, Konventionen festgelegt, denen Softwareentwickler unbedingt folgen sollten (siehe dazu [Msc2]).

3 Security Analysis Tool 2 (SAT 2)

In diesem Kapitel wird das *Security Analysis Tool 2* vorgestellt, das im Rahmen des Projektes *SAT2* am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) entwickelt wurde.

Eine Weiterentwicklung des *SAT1* Prototyps (siehe Kapitel 1) war aus vielen Gründen wünschenswert und auf Grund der zahlreichen Neuerungen, vor allem am Betriebssystemsektor, notwendig geworden. So erfolgte im Winter 2000/2001 der Projektstart des *SAT2* Projektes. Die o.a. Neuerungen, welche das zum damaligen Zeitpunkt noch relative neue Windows 2000 Betriebssystem mit sich brachte, und das Vorhaben, diese in die Sicherheitsanalyse einzubinden, legten eine Unterteilung des Tools in verschiedene Komponenten und dem zufolge auch eine Aufgabenverteilung an mehrere Entwickler nahe. Zur großen Freude des gesamten Projektteams, aber auch des Institutes FIM, erhielt der Projektleiter Dipl.-Ing. Hörmanseder nach einigen Gesprächen mit *MSR – Microsoft Research Cambridge (UK)* [Msr] eine Zusage auf finanzielle Unterstützung.

Die Entwicklung der einzelnen Komponenten wurde von mehreren Studenten (siehe Kapitel 3.2) der Johannes Kepler Universität Linz übernommen und stellt jeweils den praktischen Teil ihrer Diplomarbeiten dar.
Der Aufgabenbereich des Autors dieser Arbeit setzt sich aus zwei Teilen zusammen. Einer davon ist die Implementierung eines *Scanners* zur Benutzer- und Gruppenanalyse. Dieser Teil bestand zu einem Gutteil aus einer Übernahme bzw. Updates der in *SAT 1.0* verwendeten Funktionalität (siehe dazu auch [Han]). Der zweite und wesentlich komplexere Teil ist die Implementierung einer Visualisierungsapplikation, die einerseits die Analyse der Daten aus den sicherheitsrelevanten Bereichen eines Windows 2000 Netzwerks, und andererseits die Darstellung der Analyseergebnisse umfasst.

Den Hauptteil dieses Kapitels bildet natürlich die detaillierte Beschreibung der *SAT2* Komponenten, welche vom Autor selbst entwickelt wurden. Um dem Leser aber ein Gesamtbild vermitteln zu können, sollen auch die Aufgabenbereiche der übrigen Teammitglieder beschrieben werden, die – um dies vorwegzunehmen – die Ausgangsbasis für die Sicherheitsanalyse liefern.

3.1 Motivation und Intention

In den Zuständigkeitsbereich eines Netzwerk-Administrators fällt u.a. die Verwaltung von Benutzerberechtigungen in einem Netzwerk. Diese Aufgabe birgt aus mehreren Gründen eine nicht zu unterschätzende Komplexität in sich.
Berechtigungen können in Windows 2000 Netzwerken für faktisch alle verfügbaren Ressourcen vergeben werden. Typischerweise stellen diese Ressourcen z.B. Objekte des Dateisystems dar, also Verzeichnisse und Dateien. Berechtigungen können aber auch für Peripheriegeräte, wie z.B. Drucker, vergeben werden. Die Rechtevergabe erfolgt dabei auf Objektebene, d.h. für jedes einzelne dieser Objekte wird festgelegt, wer mit welchen Berechtigungen darauf Zugriff erhält.
Die kontinuierlich steigenden Anforderungen an die (Netzwerk-) Sicherheit und die ständige Erweiterung der sicherheitsrelevanten Bereiche eines Systems erhöhen natürlich auch die Komplexität der Aufgabenstellung und fordern immer mehr die Aufmerksamkeit des Administrators. Das Ziel, welches mit dem SAT-Projekt verfolgt wird, ist den Administrator bei seiner Arbeit zu unterstützen, indem ihm durch ein Sicherheitsanalyse-Tool kritische Punkte bzw. mögliche Sicherheitslöcher im Netzwerk aufgezeigt werden.

Standard-Werkzeuge, die mit Windows 2000 Server mitgeliefert werden, besitzen nicht die Funktionalität, die Sicherheitseinstellungen einer großen Zahl von Objekten in übersichtlicher Form darzustellen. Die einzige Möglichkeit für einen Administrator besteht darin, die Berechtigungen jeweils für ein ausgewähltes Objekt zu betrachten. Diese Standard-Werkzeuge, wie beispielsweise *Windows-Explorer* für Objekte im Dateisystem, *Registry-Editor* für die Windows Registrierungsdatenbank oder *ADSI-Edit* im Bereich *Active Directory*, bieten leider nur diese zuvor angesprochene *Objekt-zentrierte Sicht* auf die Benutzerberechtigungen.

Im Gegensatz dazu wurde mit *SAT* ein Sicherheitsanalyse-Tool entwickelt, das dem Administrator eine *Benutzer-zentrierte Sicht* auf die Berechtigungen ermöglicht. Diese interaktive Applikation beantwortet Fragestellungen wie z.B. „Wo (im gesamten System) hat ein bestimmter Benutzer Rechte?“, in Form einer übersichtlichen Auflistung aller Objekte, welche diese Bedingung erfüllen. Dadurch erhält der Administrator jederzeit einen Überblick über die Rechtestrukturen innerhalb des Netzwerkes. Er kann damit

leichter sicherheitskritische Bereiche für die betrachteten Benutzer erkennen und so das System besser absichern.

Der Aufwand, den die Kontrolle der Benutzerberechtigung unter Verwendung der Windows Standard-Werkzeuge mit sich bringt, soll durch Abbildung 3.1 verdeutlicht werden. Am Beispiel des *NT Dateisystems* werden die zahlreichen Einstellungsmöglichkeiten der Zugriffsrechte für einzelne Objekte aufgezeigt. Diese Einstellungen können für jedes Objekt im *NTFS* vorgenommen und müssen folglich auch auf diesem Weg kontrolliert werden.

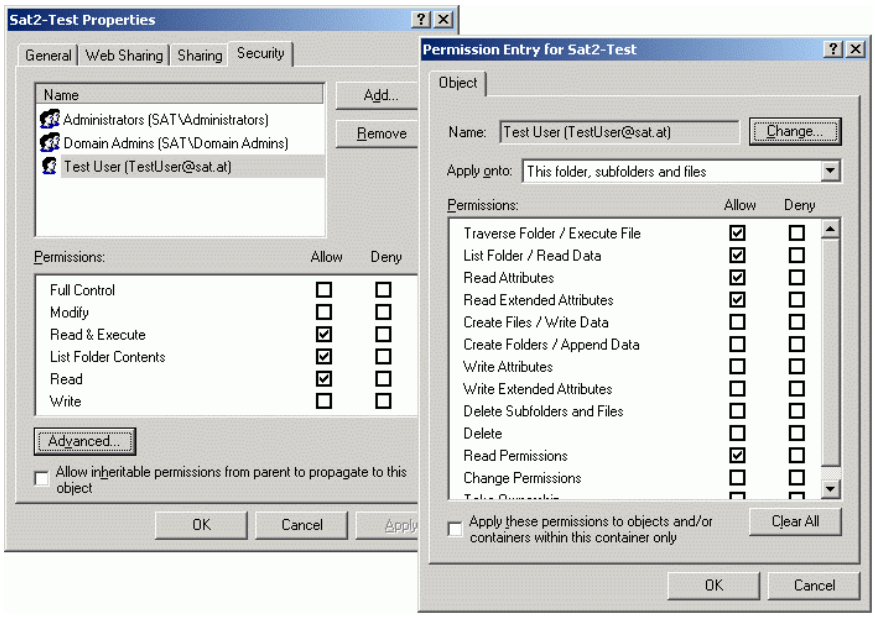


Abbildung 3.1: Einstellungsmöglichkeiten für Benutzerberechtigungen

Abbildung 3.1 zeigt die (möglichen) Sicherheitseinstellungen für einen Ordner mit der Bezeichnung *Sat2-Test* im Dateisystem. Die *Objekt-zentrierte* Sicht, welche die Standard-Werkzeuge auf die Zugriffsrechte bieten, erfordert also die Kontrolle jedes einzelnen Objektes im System.

Man stelle sich vor, wie viel Zeit alleine die Kontrolle der Dateien in seinem „Eigene Dateien“ Verzeichnis in Anspruch nimmt und wie leicht einem dabei ein Fehler unterlaufen kann. Ein Netzwerk-Administrator kann sich solche Fehler nicht erlauben und wenn man bedenkt, dass auf einem „kleinen“ Server optimistisch geschätzt rund 100.000 Dateien gespeichert sind, ist sogar für erfahrene Administratoren, abgesehen

vom Zeitaufwand, auch die Wahrscheinlichkeit des Auftretens eines Fehlers keineswegs gering.

Kommen zusätzlich noch Verzeichnisdienste wie *Active Directory Services* im Netzwerk zum Einsatz, steigen die zuvor angesprochenen Schwierigkeiten um ein Vielfaches. Die Komplexität dieser Technologie ist auf Grund des Umfangs des *Active Directory* und des dabei verfügbaren, sehr leistungsfähigen Berechtigungskonzeptes wesentlich höher als im Dateisystem.

Eine Lösung dieser Problematik wird durch den Einsatz des *Security Analysis Tools* erreicht. *SAT2* analysiert hier (im Gegensatz zum älteren *SAT 1.0*) auch das *Active Directory* und versucht damit auch, für diese komplexen Zugriffsrechte eine übersichtliche Darstellung zu finden.

Einem Netzwerk-Administrator kann mit *SAT2* folgende sicherheitsrelevante Bereiche in die Sicherheitsanalyse „seines“ Windows 2000 Netzwerkes einbeziehen:

- ***Benutzer und Gruppen Mitgliedschaften***
- ***Active Directory***
- ***NTFS 5***
- ***Registry***

Eine weiter Neuerung bietet die Visualisierungs-Applikation, die in *SAT2* als *MMC Snap-in* entwickelt wurde, und die gänzlich neu implementierte Datenbankschnittstelle.

Um trotz der zu erwartenden großen Datenmenge, die im Rahmen einer Sicherheitsanalyse verarbeitet werden muss, eine angemessene Performance zu erreichen, wurde die Teilung der Applikation in mehrere Komponenten, sowie die Zwischenspeicherung der Daten in einer Datenbank beibehalten.

3.2 Das SAT2 Team

Im folgenden Abschnitt sollen die Mitglieder des Projektteams namentlich erwähnt und ein Überblick über ihre Aufgabenbereiche gegeben werden. Des weiteren wird hier auf die verschiedenen Publikationen bzw. (Diplom-)Arbeiten der einzelnen Personen verwiesen.

- ***Dipl.-Ing. Hörmanseder Rudolf***

Die technische Projektleitung blieb selbstverständlich im Aufgabenbereich von Hrn. Hörmanseder. Seine Ideen, sowie die Grundkonzepte des Sicherheitsanalyse-Tools, können in [Hoe1] bzw. [Hoe2] nachgelesen werden.

- ***Achleitner Michael***

Die Analyse von *NTFS 5* und *Windows-Registry* wurde von Hrn. Achleitner übernommen. Informationen und Details über seine Arbeit und Sicherheit in Windows 2000 in Bezug auf *NTFS* und *Registry* werden in [Ach] präsentiert werden.

- ***Helml Thomas***

Der Zuständigkeitsbereich von Hrn. Helml wurde die Integration des *Active Directory* in die Sicherheitsanalyse und die Neuimplementierung der Datenbankschnittstelle, welche auf Grund der Änderung des Datenbanksystems notwendig war. Die Details seiner Arbeit wurden in [Hel] beschrieben.

- ***Schmitzberger Heinrich***

Das „jüngste“ Mitglied wird durch Hrn. Schmitzberger repräsentiert. Er ist dem Team erst im Winter 2001/2002 beigetreten. Seine Aufgabe wurde es, den Controller zu überarbeiten und in Form eines *MMC Snap-ins* zu implementieren. Die Ergebnisse seiner Implementierung werden nach Abschluss der Arbeiten in [Sch] nachzulesen sein.

- ***Zarda Gerald***

Der Autor dieser Arbeit übernahm die Benutzer- und Gruppenanalyse (siehe Kapitel 3.6.1), sowie die Entwicklung der Visualisierungs-Applikation als *Snap-in* für die *Microsoft Management Console (MMC)* (siehe Kapitel 3.7).

3.3 Architektur

Das *Security Analysis Tool 2 (SAT2)* setzt sich aus zwei Hauptkomponenten zusammen, die einerseits Daten in eine gemeinsame genutzte Datenbank schreiben, und andererseits wesentliche Informationen daraus beziehen.

- ***Datensammler-Komponente (Scanner)***
  Liest Benutzerberechtigungen für die sicherheitsrelevanten Bereiche aus und speichert diese in der Datenbank.

- ***Visualisierungs-Komponente***
  Zuständig für die Analyse der Benutzerberechtigungen und übersichtliche Darstellung derselben, auf Basis der Daten, die von den Datensammlern gespeichert wurden.

Das Konzept der Zwischenspeicherung der Daten bzw. Datenarchivierung ist aus mehreren Gründen vertretbar. Ein Administrator ist dadurch im Stande, das Netzwerk zu einem beliebigen Zeitpunkt zu scannen und zu einem späteren Zeitpunkt eine Analyse der Benutzerberechtigungen durchzuführen. Auf diese Weise erreicht man eine gewisse Unabhängigkeit von Netzwerkressourcen und eine Entlastung des Netzwerkes. So kann z.B. eine Auswertung auch dann erfolgen, wenn der zu untersuchende Computer nicht eingeschaltet oder vorübergehend nicht über das Netzwerk erreichbar ist. Durch die Datenarchivierung können überdies Vergleiche zu früheren Auswertungen angestellt werden.

Die Datensammler-Komponente, die von einem zentralen *Controller*-Programm (siehe Kapitel 3.5) verwaltet wird, wurde in mehrere Subkomponenten gegliedert. Jede dieser Subkomponenten ist für die Analyse eines bestimmten Bereichs im Netzwerk zuständig und schreibt die dabei anfallenden Daten direkt in die Datenbank. Dieses Konzept verleiht dem Datensammler ein hohes Maß an Flexibilität in Bezug auf die Konfigurierbarkeit, da jeder der verschiedenen Bereiche unabhängig von den anderen analysiert werden kann.

Im Projekt *SAT2* wurde jedem Teammitglied ein spezieller Aufgabenbereich zugeteilt. Dadurch konnte sich jedermann auf sein Gebiet spezialisieren und dafür einen quasi eigenständigen *Scanner* entwickeln.

Die Datensammler-Komponente des *Security Analysis Tool 2 (SAT2)* besteht aus folgenden Subkomponenten, die jeweils die Analyse eines bestimmten sicherheitsrelevanten Bereichs in einem Windows 2000 Netzwerk ermöglichen:

- ***Benutzer- u. Gruppenanalyse:*** Liefert Basisdaten für spätere Auswertungen über Computer, Benutzer und Gruppen sowie über deren Mitgliedschaften bzw. Gruppenzugehörigkeit (siehe dazu Kapitel 3.6.1).

- ***Active Directory Analyse:*** Liefert Informationen über *Active Directory* Objekte und Schema (siehe dazu [Hel/S.72 ff] und Kapitel 3.6.2)

- ***NTFS Analyse:*** Liefert Informationen über das Dateisystem der zu untersuchenden Computer (siehe dazu [Ach] und Kapitel 3.6.3).

- ***Registry Analyse:*** Liefert Informationen über Sicherheitseinstellungen in der Windows Registrierungsdatenbank (siehe dazu [Ach] und Kapitel 3.6.3).

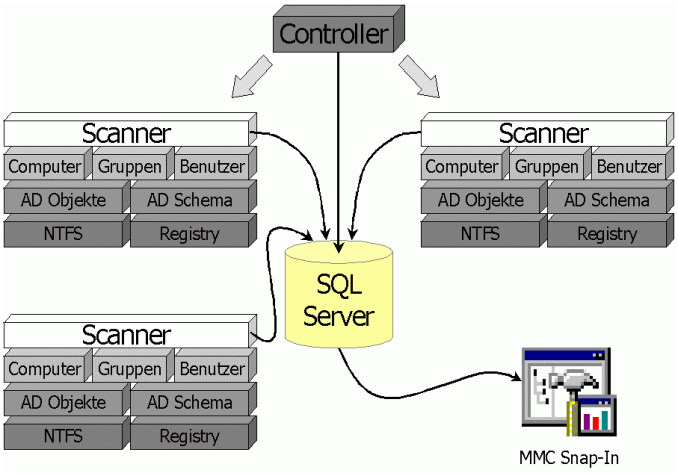


Abbildung 3.2: Architektur SAT2

SAT2 wurde so konzipiert, dass die einzelnen Datensammler-Komponenten als Dienst (*Service*) parallel auf mehreren Rechnern im Netzwerk laufen können. Um jedoch den Datenbank-*Server* zu entlasten und die Netzwerklast zu reduzieren werden die Subkomponenten eines Datensammlers sequentiell ausgeführt. Zuständig für das korrekte Starten der *Services* ist das zentrale *Controller*-Programm, das jede einzelne Instanz des Datensammlers (*Scanners*) verwaltet und überwacht.

Der zweite (Haupt-) Teil des *SAT2* Systems ist die Visualisierungs-Komponente. Diese ermöglicht eine interaktive Auswertung der Informationen, die von den Datensammlern in der Datenbank gespeichert wurden und sorgt für eine aussagekräftige und übersichtliche Darstellung der Analyseergebnisse. Die Wahl eines geeigneten Mediums zur Visualisierung fiel auf die *Microsoft Management Console (MMC)*, für die ein *Snap-in* entwickelt wurde, das die angestrebte *Benutzer-zentrierte Sicht* auf die Benutzerberechtigungen realisieren soll. Die Implementierung dieser Komponente spiegelt den praktischen Teil dieser Diplomarbeit wieder und wird ausführlich in Kapitel 3.7 beschrieben.

3.4 Datenbank

Zur Zwischenspeicherung und Archivierung der Daten aus den sicherheitsrelevanten Bereichen des Netzwerks wird der *Microsoft SQL Server 2000* verwendet. Die Entscheidung für einen DB-Server wurde aus Performance-Gründen getroffen, da das in *SAT 1.0* verwendete *MS Access* relativ schnell an seine Belastungsgrenzen stößt. Auch wollte das *SAT2* Team keinen Daten-Konzentrator – wie in der Vorgängerversion – in das Konzept integrieren.

Dieser Umstieg erforderte jedoch eine Neuimplementierung der Datenbankschnittstelle, welche von Hrn. Helml durchgeführt wurde. Der Zugriff auf den Datenbank-Server wurde mittels *ActiveX Data Objects (ADO)* realisiert (siehe dazu [Hel/S.76 ff.]). Diese Datenbankschnittstelle bietet derzeit die notwendige Basisfunktionalität, stellt aber den ersten Prototyp dar und muss in Zukunft verstärkt optimiert werden.

Zusätzlich wurden – trotz der höheren Leistungsfähigkeit des *SQL Servers* – in die einzelnen Datensammler-Komponenten *Caching-Routinen* integriert, um durch eine Re-

duktion der Datenbankzugriffe den *SQL Server* zu entlasten und so eine Steigerung der Performance zu erreichen (siehe Kapitel 3.6.2).

Allerdings muss an dieser Stelle darauf hingewiesen werden, dass beim Entwurf der Datenbank wesentliche Designrichtlinien außer Acht gelassen werden mussten. So unterliegt die Datenbank beispielsweise (leider) keiner Normalform. Darüber hinaus werden wesentliche Daten teilweise redundant gespeichert, um bereits im ersten Prototyp zusätzliche Zugriffe bei der Auswertung einzusparen.

Das *SAT2* Team ist sich darüber im Klaren, dass in diesem Bereich noch viele Optimierungen durchgeführt werden können und auch müssen. Diese Optimierungen werden aber erst in künftigen *SAT2* Versionen stattfinden. In Hinblick auf eine bessere Performance des *SAT2 Snap-ins*, genauer gesagt um die Wartezeit während der Durchführung einer Analyse zu verkürzen, geht der Trend immer mehr in die Richtung, dass ein Großteil der Daten direkt in den Datensatz eines Objektes (d.h. als jeweils ein Eintrag in der DB-Tabelle „objects“) gespeichert werden.

Da die im Rahmen dieser Diplomarbeit beschriebenen *Scanner*-Komponenten in die Datenbank schreiben und die Visualisierungs-Komponente zur Auswertung ausschließlich auf Daten aus der Datenbank zurückgreift, soll folgender Abschnitt dem Leser einen Überblick verleihen, welche Tabellen in der *SAT2* Datenbank verwendet und mit welchen Daten diese gefüllt werden. Die Auflistung der Tabellen wird alphabetisch durchgeführt und hat keinerlei Bedeutung in Bezug auf Reihenfolge beim Schreiben, Größe oder Wesentlichkeit der Daten und dgl.

3.4.1 Tabelle „ace“

Benutzerberechtigungen werden in Windows Systemen anhand spezieller Zugriffssteuerungslisten (ACLs) für Objekte vergeben. Diese Liste setzt sich aus einer variablen Anzahl von Zugriffssteuerungseinträgen zusammen, die als *Access Control Entries (ACEs)* bezeichnet werden. Durch diese Einträge wird festgelegt, wer mit welchen Berechtigungen auf das Objekt zugreifen darf. In der Tabelle „ace“ werden diese Einträge mit einem Verweis auf die ihnen zugehörige ACL gespeichert.

Attribut	Datentyp	Beschreibung
acl_id	int	ID der ACL, welcher der ACE entnommen wurde
sid	varchar(SAT_SID_LEN)	Sicherheits-ID des Benutzers bzw. einer Gruppe
mask	tinyint	Zugriffsrechte
type	tinyint	Typ des ACE
flags	tinyint	Vererbungsstrategie
objType	varchar (SAT_GUID_LEN)	Typ des Objektes
inheritedObjectType	varchar (SAT_GUID_LEN)	Vererbter Objekttyp
fromStdSecurity	bit	Zugriffsrechte entsprechen den Standardeinstellungen

Tabelle 3.1: DB-Tabelle „ace“

Diese Tabelle wird von *ADS-Scanner* und *NTFS/REG-Scanner* beschrieben und von der Visualisierungs-Komponente u.a. zum Ermitteln der Benutzerberechtigungen verwendet.

3.4.2 Tabelle „acl“

Die sogenannte *Access Control List (ACL)* enthält sämtliche Zugriffsteuerungseinträge, aus denen sich die Benutzerberechtigungen für ein Objekt zusammensetzen. Zu diesem Zweck wird die ACL in einem Windows System immer gemeinsam mit dem Objekt gespeichert. Daraus folgt aber, dass für jedes einzelne Objekt im System eine ACL existiert und demnach die zu erwartende Datenmenge relativ hoch ist. Aus diesem Grund werden ACLs in der *SAT2* Datenbank mit einer ID versehen und nur einmal in der Datenbank abgelegt. Um den Kontext zu wahren, wird diese ID sowohl zu jedem ACE-Eintrag als auch zu jedem Objekt in der Datenbank gespeichert. Anmerkung: Auch im *NTFS 5* findet man diese Art der ACL/ACE-Verwaltung vor.

Attribut	Datentyp	Beschreibung
acl_id	int	ID der ACL, welcher der ACE entnommen wurde
hash	int	Hash-Wert um DB-Zugriffe zu reduzieren

Tabelle 3.2: DB-Tabelle „acl“

Um die Anzahl der Datenbankzugriffe zu minimieren, wird ein numerischer Hash-Wert generiert. Dieser Hash-Wert berechnet sich aus der Addition aller *Access Masks* sämtlicher Zugriffssteuerungseinträge (ACEs) einer ACL. Die Einträge dieser Tabelle werden von den *Scannern* generiert, sind aber für die Auswertung der Berechtigungen nicht von Bedeutung.

3.4.3 Tabelle „computers“

Diese Tabelle enthält Informationen über sämtliche Computer des Netzwerkes, die analysiert wurden. Wesentlich daran ist, dass der SID einer *Workstation* unterschiedlich zu dem eines *Domain-Controllers* zu behandeln ist.

Attribut	Datentyp	Beschreibung
computer_id	int	Datenbank-ID des Computers
localSid	varchar(SAT_SID_LEN)	Sicherheits-ID einer Workstation
domainSid	varchar(SAT_SID_LEN)	Sicherheits-ID eines Domain-Controllers
netBiosName	varchar(SAT_NETBIOSNAME_LEN)	Net-BIOS Name
dnsHostName	varchar(SAT_DNSHOSTNAME_LEN)	DNS Host Name
dnsDomainName	varchar(SAT_DOMAINNAME_LEN)	Name der Domain, der der Computer angehört
fqDnsName	varchar(SAT_FQDNSNAME_LEN)	Fully Quallified DNS Name
version	int	Versionsnummer des Security-Scans

Tabelle 3.3: DB-Tabelle „computers“

Die Daten dieser Tabelle werden durch den *CGU-Scanner* ermittelt und als Basisdaten für die Auswertung verwendet.

3.4.4 Tabelle „groups“

Diese Tabelle enthält sämtliche Informationen über Gruppen, die für die Auswertung der Berechtigungen und zur Darstellung der Ergebnisse relevant sind. Wie die Tabelle „computers“ wird sie vom *CGU-Scanner* gefüllt und natürlich von der Visualisierungs-Komponente verwendet.

Attribut	Datentyp	Beschreibung
sid	varchar(SAT_SID_LEN)	Sicherheits-ID der Gruppe
name	varchar(SAT_GROUPNAME_LEN)	Name der Gruppe
comment	varchar(SAT_GROUPCOMMENT_LEN)	Kommentar
lg_flag	varchar(1)	Bezeichnet eine lokale oder globale Gruppe
computer_id	int	Computer, auf dem die Daten ausgelesen wurden
version	int	Versionsnummer des Security-Scans

Tabelle 3.4: DB-Tabelle „groups“

3.4.5 Tabelle „membership“

Diese Tabelle gibt Aufschluss über die Mitgliedschaften bzw. Gruppenzugehörigkeit von Benutzern und Gruppen in einem Netzwerk.

Gespeichert werden jedoch nur die Sicherheits-IDs der Mitglieder und zugehöriger Gruppen.

Attribut	Datentyp	Beschreibung
sid	varchar(SAT_SID_LEN)	Sicherheits-ID des Mitglieds (Benutzers bzw. Gruppe)
memberOfSid	varchar(SAT_SID_LEN)	Sicherheits-ID der zugehörigen Gruppe
isSidHistory	bit	Bezeichnet einen Eintrag aus der SID-History
version	int	Versionsnummer des Security-Scans

Tabelle 3.5: DB-Tabelle „membership“

Diese Tabelle wird hauptsächlich vom *CGU-Scanner* beschrieben. Unter gewissen Umständen kann aber auch der *ADS-Scanner* Werte darin speichern. Dies ist genau dann der Fall, wenn ein Benutzer im *Active Directory* durch mehrere *SIDs* identifiziert werden kann und wirkt sich auf das Attribut „isSidHistory“ aus (siehe dazu Kapitel 3.6.1.4 und [Hel/S.112 f.]). Die Visualisierungs-Komponente verwendet diese Tabelle z.B. um die „effektiven“ Rechte eines Benutzers zu bestimmen, die er durch seine Gruppenzugehörigkeit erhalten kann.

3.4.6 Tabelle „objects“

Diese Tabelle enthält detaillierte Informationen über alle Objekte aus den sicherheitsrelevanten Bereichen eines Netzwerkes und stellt damit den Mittelpunkt aller Auswertungen dar. Diese Informationen bilden zusammen mit den Daten aus der „ace“-Tabelle die Basis für eine Auswertung durch die Visualisierungs-Komponente.

Attribut	Datentyp	Beschreibung
obj_id	int	Datenbank-ID des Objektes
objName	varchar(SAT_OBJNAME_LEN)	Name des Objektes
objGuid	varchar(SAT_GUID_LEN)	GUID des Objektes
comp_id	int	Computers, von dem das Objekt ausgelesen wurde
acl_id	int	ID der ACL des Objektes
objTypeGuid	varchar(SAT_GUID_LEN)	GUID des Objekttyps
amount	int	Anzahl (bei Komprimierung)
version	int	Versionsnummer des Security-Scans
parent	int	ID des Vaterobjektes
path	varchar(SAT_COMPR_PATH_LEN)	Komprimierter Pfad
breaksStdSecurity	bit	Signalisiert einen Bruch in der Standard-Security
breaksInheritance	bit	Signalisiert einen Bruch in der Vererbung
date	char(20)	Erzeugungsdatum des Objektes
fullPath	varchar(SAT_FULL_PATH_LEN)	Vollständiger Pfad

Tabelle 3.6: DB-Tabelle „objects“

3.4.7 Tabelle „objTypes“

Anhand dieser Tabelle können Informationen über den Typ eines Objektes abgefragt werden, der in der „objects“-Tabelle als *GUID* gespeichert wird. Wird in der Visualisierungs-Komponente anstatt des Objekttyp-*GUIDs* sein Name angezeigt, trägt dies unbestreitbar zum besseren Verständnis der Auswertung bei. Gefüllt wird diese Tabelle ausschließlich vom *ADS-Scanner*

Für Objekte im Dateisystem existiert keine äquivalente Beschreibung, es sollen aber aus Konsistenzgründen die folgenden „selbst-generierten“ Objekt-Typen hier mit aufgenommen werden.

- ***Verzeichnis (Directory)***
- ***Datei (File)***
- ***Registry-Key***

Attribut	Datentyp	Beschreibung
objTypeGuid	varchar(SAT_GUID_LEN)	GUID des Objekttyps
compID	int	Computer auf dem der Objekttyp ausgelesen wurde
TypeName	varchar(SAT_TYPENAME_LEN)	Name des Objekttyps
objectClass	char(1)	Objektklasse
defaultSec	int	Signalisiert ob Obj.Klasse der Std.Security entspricht
version	int	Versionsnummer des Security-Scans

Tabelle 3.7: DB-Tabelle „objTypes“

3.4.8 Tabelle „sidRef“

Diese Tabelle wurde entworfen, um bei späteren Auswertungen feststellen zu können, wo ein Subjekt (z.B. User oder Gruppe oder auch Computer) im ADS angelegt wurde. Darin werden alle Objekte des *Active Directory*, die über einen *Security Identifier (SID)* verfügen, mit einer Referenz auf das Vaterobjekt gespeichert.

Attribut	Datentyp	Beschreibung
sid	varchar (SAT_SID_LEN)	Sicherheits-ID eines Objektes im AD
objRef	int	Objekt ID des zugehörigen Vaterobjektes
version	int	Versionsnummer des Security-Scans

Tabelle 3.8: DB-Tabelle „sidRef“

3.4.9 Tabelle „users“

Diese Tabelle enthält alle wesentlichen Informationen über die Benutzer bzw. Benutzerkonten eines Netzwerkes. Da Auswertungen in der Regel für Benutzer durchgeführt werden, bildet die vom *CGU-Scanner* beschriebene Tabelle die erste Ausgangsbasis für eine Analyse der Benutzerberechtigungen.

Attribut	Datentyp	Beschreibung
sid	varchar(SAT_SID_LEN)	Sicherheits-ID des Benutzers
accountName	varchar(SAT_USERACCNAME_LEN)	Bezeichnung des Benutzerkontos
fullName	varchar(SAT_USERFULLNAME_LEN)	Vollständiger Name des Benutzers
comment	varchar(SAT_USERCOMMENT_LEN)	Kommentar
lg_flag	varchar(1)	Bezeichnet einen lokalen oder globalen Benutzer
computer_id	int	Computer, auf dem die Daten ausgelesen wurden
version	int	Versionsnummer des Security-Scans

Tabelle 3.9: DB-Tabelle „users“

3.5 Der Controller/Master

Wie aus Kapitel 3.3 hervorgeht, verfügt das *SAT2* System über eine *Controller*-Komponente, welche die Steuerung und Kontrolle der Datensammler-Komponenten übernimmt. Dabei dient der *Controller* auch als Benutzerschnittstelle. Dadurch wird dem Anwender die Möglichkeit geboten, die Datensammler-Komponenten frei zu konfigurieren und somit jeden *Security-Scan* seinen Bedürfnissen oder den aktuellen Gegebenheiten anzupassen. Wünscht der Anwender beispielsweise nur eine Sicherheitsanalyse im *NT Dateisystem* durchzuführen, so kann dies durch entsprechende Einstellungen mit Hilfe des *Controller*-Programms festgelegt werden.

Die Standard-Konfiguration der Datensammler, sowie die vom Anwender spezifizierten Konfigurationsdaten, werden unter folgendem Schlüssel in der *Windows-Registry* gespeichert:

- **`HKEY_LOCAL_MACHINE\SOFTWARE\Fim\Sat2`**

Die im Folgenden angeführten *Registry*-Schlüssel und Werte werden zur Konfiguration der Datensammler-Komponenten verwendet und können vom Anwender nach belieben geändert werden. Zusätzlich werden sowohl die Standardwerte als auch zulässige Wertebereiche angeführt.

- ***adsCompression:*** gibt die Komprimierungsstufe für *Active Directory* Analysen an; zulässige Werte derzeit sind: 0 (keine Komprimierung) bis 3 (maximale Komprimierung); Default-Wert: 0
- ***cacheLimit:*** gibt die Größe des *Caches* an, der in den *Caching-Routinen* (z.B. maximale Anzahl von *ACLs* im *Cache*) verwendet wird; Default-Wert: 300
- ***creationTime:*** gibt an, welches Objekt-Datum zu den Objekten in der Datenbank gespeichert wird; zulässige Werte: 0 (Erzeugungsdatum) oder 1 (Datum der letzten Änderung); Default-Wert: 0
- ***dbServer:*** gibt den Namen bzw. die (Netzwerk-) Adresse des SQL-Servers an, der die analysierten Daten speichert; zulässige Werte: Net-BIOS Name, DNSName oder IP-Adresse; Default-Wert: localhost

- ***ntfsCompression:*** gibt die Komprimierungsstufe für *NTFS* Analysen an; derzeit zulässige Werte: 0 (keine Komprimierung) bis 3 (maximale Komprimierung); Default-Wert: 0
- ***ntfsEndpoints:*** gibt Verzeichnisse an, die bei einer *NTFS* Analyse nicht berücksichtigt werden sollen; zulässige Werte: Windows-Verzeichnisnamen; Default-Wert: c:\winnt
- ***ntfsStartpoints:*** gibt Laufwerke oder Verzeichnisse an, die als Startpunkte bei einer *NTFS* Analyse verwendet werden; zulässige Werte: Laufwerksbezeichnungen oder Verzeichnisnamen; Default-Wert: c:\
- ***scanAdsConfig:*** gibt an, ob bei einer *Active Directory* Analyse der *Configuration-Container* mit einbezogen werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 0
- ***scanAdsDomain:*** gibt an, ob bei einer *Active Directory* Analyse der *DomainNC-Container* mit einbezogen werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 0
- ***scanAdsSchema:*** gibt an, ob eine *Active Directory Schema* Analyse durchgeführt werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 0
- ***scanAdsSchemaRights:*** gibt an, ob bei einer *Active Directory* Analyse der *Schema-Container* mit einbezogen werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 0
- ***scanNtfs:*** gibt an, ob eine Analyse des *NT Dateisystems* durchgeführt werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 1
- ***scanRegistry:*** gibt an, ob eine Analyse der *Windows-Registry* durchgeführt werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 0
- ***scanUsers:*** gibt an, ob eine Benutzer u. Gruppenanalyse durchgeführt werden soll; zulässige Werte: 0 (nein) oder 1 (ja); Default-Wert: 1

Diese Werte werden von der derzeitigen Version des *Controllers* gesetzt, um die Datensammler-Komponente zu konfigurieren und zu aktivieren (Anmerkung: neue *Registry*-Werte können nach Bedarf hinzugefügt werden). Anschließend werden jeweils Instanzen der als *Services* implementierten *Security-Scanner* auf den/die zu untersuchenden Computer kopiert und dort gestartet. Somit können die Datensammler parallel auf verschiedenen Computern im Netzwerk ausgeführt werden. Aus Performance-Gründen

erfolgt der interne Ablauf einer *Scanner*-Instanz jedoch sequentiell, d.h. die sicherheitsrelevanten Bereiche werden jeweils nacheinander analysiert.

Eine weitere Aufgabe, für die der *Controller* zuständig ist, besteht aus dem Erstellen der *SAT2* Datenbank und dem Anlegen der zugehörigen Tabellen, die in Kapitel 3.4 im Überblick beschrieben wurden. Der Dokumentation der einzelnen *Scanner*-Komponenten können nähere Informationen darüber entnommen werden. Folgende Übersicht enthält nochmals alle Tabellen, die in der *SAT2* Datenbank vom *Controller* erstellt werden:

- ***Tabelle „ace“***
- ***Tabelle „acl“***
- ***Tabelle „computers“***
- ***Tabelle „groups“***
- ***Tabelle „membership“***
- ***Tabelle „objects“***
- ***Tabelle „objTypes“***
- ***Tabelle „sidRef“***
- ***Tabelle „users“***

Nachdem die *SAT2* Datenbank vollständig erstellt wurde, jedoch bevor die *Security-Scanner* starten, werden vom *Controller* – falls erforderlich – Basisdatensätze in die Datenbank geschrieben.
Als Beispiel hierfür seien die gebräuchlichen Sicherheitskennungen (*Well-known SIDs*) angeführt (siehe dazu [Msc1/S.1413 ff.]). Der *CGU-Scanner* ist nicht in der Lage, Informationen über Benutzer oder Gruppen zu ermitteln, die anhand dieser konstanten Sicherheitskennungen identifiziert werden. Da diese *SIDs* jedoch Einfluss auf die Benutzerberechtigungen haben und in Folge dessen auch in die Auswertung miteinbezogen werden müssen, ist der *Controller* dafür verantwortlich, diese Informationen korrekt in die Datenbank zu schreiben.

HINWEIS: Der *Controller* wurde zu Testzwecken als Kommandozeilen-Applikation implementiert. Während des Entstehens dieser Diplomarbeit wird von Hrn. Schmitzberger im Rahmen eines 10-stündigen Praktikums ein *MMC Snap-in* entwickelt, das den *Controller* mit der eben beschriebenen Funktionalität realisieren soll (siehe dazu [Sch]).

3.6 Die Datensammler (*Scanner*)

Die Datensammler-Komponente des *Security Analysis Tool 2 (SAT2)* lässt sich selbst wieder in mehrere Subkomponenten unterteilen, die jede für sich spezielle Aufgaben ausführen. D.h. für jeden sicherheitsrelevanten Bereich eines Windows 2000 Netzwerkes wurde ein spezieller *Security-Scanner* implementiert, der die analysierten Daten direkt in die dafür vorgesehene SQL-Datenbank schreibt.

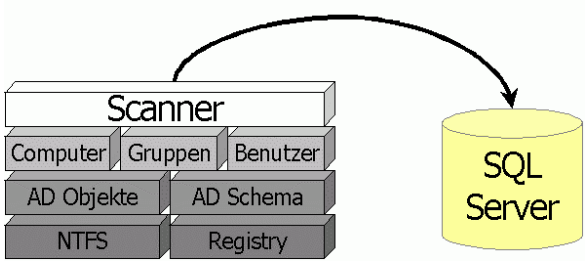


Abbildung 3.3: SAT2 Security-Scanner

Der *SAT2 Security-Scanner* liefert die Datenbasis zur Analyse der Benutzerberechtigungen in einem Netzwerk. Zu Dokumentationszwecken wird in dieser Diplomarbeit zwischen drei Hauptkomponenten eines *Security-Scanners* unterschieden:

- ***CGU-Scanner:*** Sammelt Informationen über Computer, Gruppen und Benutzer sowie Gruppenmitgliedschaften, auf den zu untersuchenden Maschinen. (siehe Kapitel 3.6.1)

- ***ADS-Scanner:*** Wird für die Sicherheitsanalyse im Active Directory eingesetzt und liefert dazu Sicherheitseinstellungen von AD Objekten und AD Schema (siehe [Hel/S.72 ff.] und Kapitel 3.6.2).

- ***NTFS/REG-Scanner:*** Liefert sicherheitsrelevante Informationen über das Dateisystem und die Registrierungsdatenbank auf ausgewählten Computern (siehe [Ach] und Kapitel 3.6.3).

Im folgenden Abschnitt werden die einzelnen Datensammler-Komponenten beschrieben. Dabei wird jedoch das Hauptaugenmerk auf den *CGU-Scanner* gelegt, der vom Autor dieser Arbeit implementiert wurde.

3.6.1 Der CGU-Scanner

Eine der Aufgaben des Autors dieser Arbeit im *SAT2* Projekt war die Implementierung einer – nicht unwesentlichen – Komponente des *Security-Scanners*. Der im folgenden Kapitel beschriebene ***C****omputer-****G****roups-****U****sers-Scanner* liefert Basisdaten, die für jede Auswertung der Benutzerberechtigungen zwangsläufig erforderlich sind. Dabei wird allerdings weniger Wert auf die Details in Bezug auf die Implementierung an sich gelegt, als auf eine allgemeine Beschreibung des Gesamtkonzeptes in Hinblick auf spätere Auswertungen.

Wie eingangs bereits erwähnt, liefert der *CGU-Scanner* Informationen über die zu analysierenden Computer, sowie über Gruppen und Benutzer, welche auf diesen Maschinen lokalisiert werden können und schreibt diese direkt in die, in der Datenbank dafür vorgesehenen Tabellen. Abbildung 3.4 zeigt die Architektur des *CGU-Scanners*.

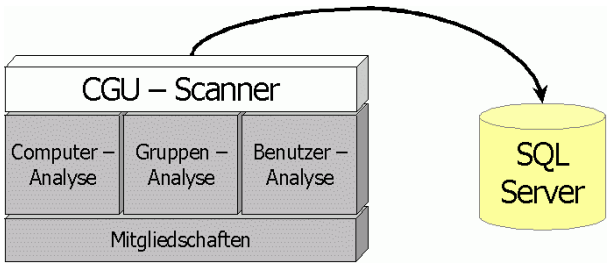


Abbildung 3.4: Architektur CGU-Scanner

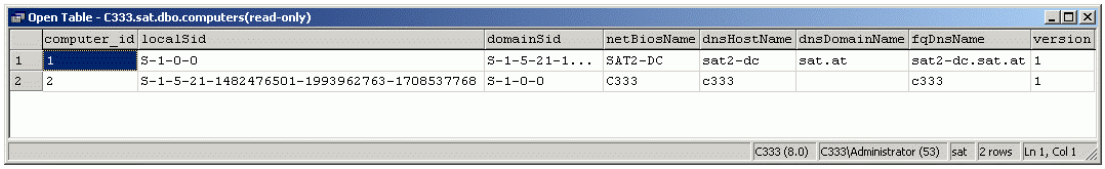
Wesentlich daran ist aber, dass die Reihenfolge der Ausführung der einzelnen Subkomponenten des *CGU-Scanners* von Bedeutung ist. Zu Beginn müssen die Computer-Informationen gespeichert werden, da für jeden gescannten Computer ein Index in der Datenbank vergeben wird, auf den später in anderen Tabellen verwiesen wird. Dies betrifft sowohl die übrigen Teile des *CGU-Scanners*, als auch den *ADS*- und *NTFS/REG-Scanner*. Anschließend wird (sequentiell) die Gruppen- und Benutzer- und Mitgliedschaften-Analyse (in dieser Reihenfolge) gestartet. Zur Unterscheidung von Daten verschiedener Analyseläufe wurde eine Versionsnummer eingeführt, die vom *Controller*

verwaltet und mit jedem Datensatz, den der *CGU-Scanner* schreibt, mitgespeichert wird.

3.6.1.1 Computer

Die Informationen, welche über Computer in der Datenbank gespeichert werden, beziehen sich hauptsächlich auf deren unterschiedliche Bezeichnungen bzw. Namen. So werden für jeden Computer der *Net-BIOS* Name, *DNS Host* Name, *DNS Domain* Name und der *Fully Qualified DNS* Name gespeichert.

Abbildung 3.5 zeigt die Tabelle „computers“ der *SAT2* Datenbank, nach einem erfolgreichen *Security-Scan*. Die Beschreibung der Attribute und Datentypen kann Kapitel 3.4.3 entnommen werden.


Open Table - C333.sat.dbo.computers(read-only)

	computer_id	localSid	domainSid	netBiosName	dnsHostName	dnsDomainName	fqDnsName	version
1	1	S-1-0-0	S-1-5-21-1...	SAT2-DC	sat2-dc	sat.at	sat2-dc.sat.at	1
2	2	S-1-5-21-1482476501-1993962763-1708537768	S-1-0-0	C333	c333		c333	1

C333 (8.0) | C333\Administrator (53) | sat | 2 rows | Ln 1, Col 1

Abbildung 3.5: Tabelle „computers“

Computer können in Windows 2000 anhand einer Domain-weit eindeutigen Sicherheitskennung (SID) identifiziert werden und fallen somit in die Kategorie der Sicherheitsprincipals (*Security-Principals*). Aus diesem Grund werden auch Computer-SIDs in der Datenbank gespeichert.

Zu beachten ist in diesem Zusammenhang allerdings, dass ein Computer u.U. über mehrere SIDs verfügen kann. Prinzipiell werden Computer-SIDs bei Installation des Betriebssystems (vom System selbst) vergeben. Dennoch besteht die Möglichkeit, dass ein Computer zusätzlich einen SID erhält und zwar genau dann, wenn er in eine Domain aufgenommen wird.
Problematisch wird dies deshalb, weil diese SIDs im programmiertechnischen Sinn unterschiedlich zu behandeln sind und zwar in Abhängigkeit vom „Typ“ des Computers.
Die Funktion zum Auslesen des Computer SIDs erhält als Eingangsparameter den *Net-BIOS* Namen des Computers. Wird diese auf einem „*stand-alone*“ Computer aufgeru-

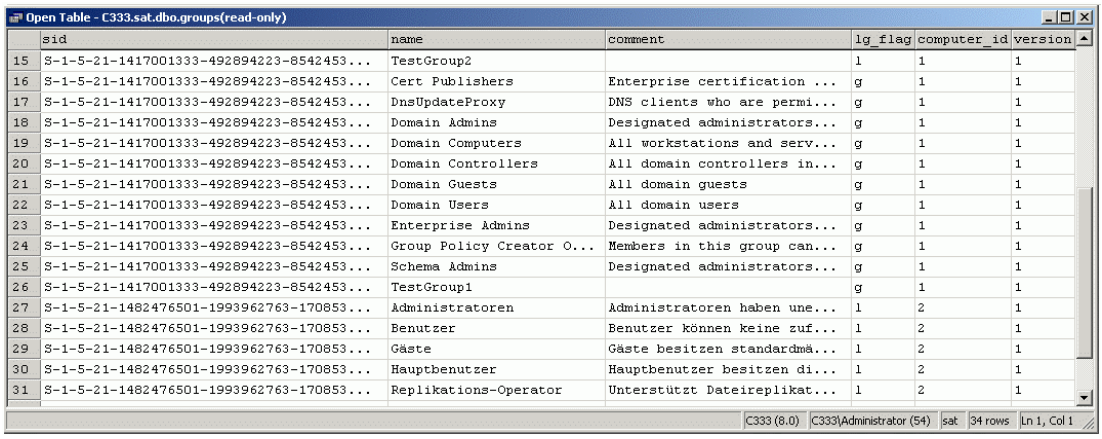
fen, so liefert sie als Rückgabewert den Computer-SID, der als „localSid“ in der Datenbank gespeichert wird.
Für *Domain-Controller* kann zwar die selbe Funktion verwendet werden, es ist jedoch unbedingt darauf zu achten, dass an den, als Eingangsparameter verwendeten *Net-BIOS* Namen ein $-Zeichen anzuhängen ist. Ansonsten würde der Versuch, den SID auszulesen, mit einer Fehlermeldung enden. Der SID eines *Domain-Controllers* wird in der Datenbank als „domainSid“ gespeichert.
„*Stand-alone*“ Computer oder Workstations, die als Mitglieder in eine Domain aufgenommen werden, verfügen allerdings über beide (!) SIDs. Deshalb mussten in der Datenbank auch zwei Felder (Attribute) vorgesehen werden, in welchen die unterschiedlichen SIDs gespeichert werden können, da sonst eine korrekte Analyse der Berechtigungen eines Computers im gesamten Netzwerk nicht möglich wäre.

3.6.1.2 Gruppen (Groups)

Windows Betriebssysteme bieten die Möglichkeit, Benutzer, welche den gleichen Sicherheitsbestimmungen unterliegen oder die gleiche Berechtigungen zur Durchführung ihrer Aufgaben benötigen, zu Gruppen zusammenzufassen. Dies erleichtert die Verwaltung eines Systems enorm, da so in einem Arbeitsschritt allen Mitgliedern dieser Gruppe die gleichen Rechte zugewiesen werden können und damit eine gewisse Konsistenz und auch Änderungsfreundlichkeit in der Rechtevergabe gewährleistet ist.

Identifiziert werden diese Sicherheitsprincipals wiederum anhand eines *Security Identifiers*, der beim Erstellen einer Gruppe durch das Betriebssystem vergeben wird. Zusätzlich können für alle Gruppen nähere Informationen zur Gruppe selbst, wie ihr Name und eine Beschreibung, spezifiziert werden.

Der Gruppen-SID sowie alle anderen Informationen, welche eine Gruppe beschreiben, werden durch den *CGU-Scanner* aus der Benutzerdatenbank des Betriebssystems ausgelesen und in der *SAT2* Datenbank gespeichert. Darüber hinaus wird ein Flag zur Unterscheidung zwischen lokalen und globalen Gruppen, sowie die ID des Computers, von dem die Gruppeninformationen ausgelesen wurden, mitgespeichert (siehe Abbildung 3.6).



Open Table - C333.sat.dbo.groups(read-only)

	sid	name	comment	lg_flag	computer_id	version
15	S-1-5-21-1417001333-492894223-8542453...	TestGroup2		l	1	1
16	S-1-5-21-1417001333-492894223-8542453...	Cert Publishers	Enterprise certification ...	g	1	1
17	S-1-5-21-1417001333-492894223-8542453...	DnsUpdateProxy	DNS clients who are permi...	g	1	1
18	S-1-5-21-1417001333-492894223-8542453...	Domain Admins	Designated administrators...	g	1	1
19	S-1-5-21-1417001333-492894223-8542453...	Domain Computers	All workstations and serv...	g	1	1
20	S-1-5-21-1417001333-492894223-8542453...	Domain Controllers	All domain controllers in...	g	1	1
21	S-1-5-21-1417001333-492894223-8542453...	Domain Guests	All domain guests	g	1	1
22	S-1-5-21-1417001333-492894223-8542453...	Domain Users	All domain users	g	1	1
23	S-1-5-21-1417001333-492894223-8542453...	Enterprise Admins	Designated administrators...	g	1	1
24	S-1-5-21-1417001333-492894223-8542453...	Group Policy Creator O...	Members in this group can...	g	1	1
25	S-1-5-21-1417001333-492894223-8542453...	Schema Admins	Designated administrators...	g	1	1
26	S-1-5-21-1417001333-492894223-8542453...	TestGroup1		g	1	1
27	S-1-5-21-1482476501-1993962763-170853...	Administratoren	Administratoren haben une...	l	2	1
28	S-1-5-21-1482476501-1993962763-170853...	Benutzer	Benutzer können keine zuf...	l	2	1
29	S-1-5-21-1482476501-1993962763-170853...	Gäste	Gäste besitzen standardmä...	l	2	1
30	S-1-5-21-1482476501-1993962763-170853...	Hauptbenutzer	Hauptbenutzer besitzen di...	l	2	1
31	S-1-5-21-1482476501-1993962763-170853...	Replikations-Operator	Unterstützt Dateireplikat...	l	2	1

C333 (8.0) | C333\Administrator (54) | sat | 34 rows | Ln 1, Col 1

Abbildung 3.6: Tabelle „groups“

Für spätere Auswertungen sind diese Daten besonders wichtig, weil einerseits die Berechtigungen einer bestimmten Gruppe im System bzw. Netzwerk abgefragt und andererseits Gruppeninformationen verständlich präsentiert werden können. (Selbst der erfahrenste Administrator kennt nicht die SIDs „seiner“ Benutzer, genauso wenig wie ein Internet-Server die IP-Adressen der Computer. Alle arbeiten mit symbolischen Namen.)

Probleme bei der Entwicklung des *CGU-Scanners* gab es allerdings bei den sogenannten *Built-in Groups*. In jedem System gibt es vordefinierter Benutzergruppen, denen Benutzer zugeordnet werden können bzw. zu denen man beim Anmelden an ein System automatisch zugeordnet wird. In folgender Tabelle werden einige dieser *Built-in Groups* angeführt (siehe dazu [Msc1/S.1413 ff.]).

SID	Name	Beschreibung
S-1-1-0	Jeder (*Everyone*)	Gruppe, die alle Benutzer einschließt (inklusive anonyme Benutzer und Gäste)
S-1-5-1	Dialup (*Dialup*)	Gruppe, die alle Benutzer enthält, die über eine Einwählverbindung am Computer angemeldet sind
S-1-5-2	Netzwerk (*Network*)	Gruppe, die alle Benutzer enthält, die über eine Netzwerkverbindung am Computer angemeldet sind
S-1-5-4	Interaktiv (*Interactive*)	Gruppe, die alle Benutzer enthält, die sich interaktiv an das System angemeldet haben
S-1-5-6	Dienst (*Service*)	Gruppe, die alle Security-Principals enthält, die sich als Dienst angemeldet haben

Tabelle 3.10: Built-in Groups und Well-known SIDs

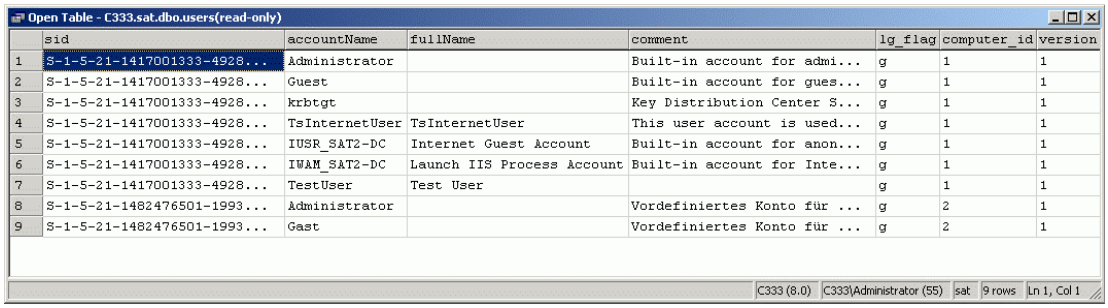
Mit den Befehlen, die zum Auslesen der Gruppeninformationen verwendet werden können, besteht keine Möglichkeit, Daten über diese vordefinierten Gruppen zu ermitteln. Da eine Mitgliedschaft in einer dieser Gruppen u.U. aber gravierenden Einfluss auf die

Rechte eines Sicherheitsprincipals haben kann, sind diese Daten jedoch zwingend erforderlich.

Einzige Lösung des Problems: der *Controller* sorgt für die Verfügbarkeit dieser Daten, indem er sie vor dem Starten des *CGU-Scanners*, in der Datenbank speichert. Auf Grund der Tatsache, dass diese Werte immer konstant bleiben, hat diese Methode keine negativen Auswirkungen auf spätere Auswertungen oder Analysen.

3.6.1.3 Benutzer (Users)

Um sich in einem Windows 2000 Netzwerk anmelden zu können, benötigt man zumindest ein Benutzerkonto (*Account*) und natürlich die entsprechenden Berechtigungen dazu. In der Regel werden diese Benutzerkonten vom Netzwerk-Administrator erstellt und zwar jeweils für einen Benutzer. Zusätzlich können Benutzer durch Angabe des vollständigen Namens und eines Kommentars näher beschrieben werden. Zur Identifikation eines Benutzers wird wiederum ein *Security Identifier (SID)* verwendet, der beim Erstellen des Kontos vom System vergeben wird.


Open Table - C333.sat.dbo.users(read-only)

	sid	accountName	fullName	comment	lg_flag	computer_id	version
1	S-1-5-21-1417001333-4928...	Administrator		Built-in account for admi...	g	1	1
2	S-1-5-21-1417001333-4928...	Guest		Built-in account for gues...	g	1	1
3	S-1-5-21-1417001333-4928...	krbtgt		Key Distribution Center S...	g	1	1
4	S-1-5-21-1417001333-4928...	TsInternetUser	TsInternetUser	This user account is used...	g	1	1
5	S-1-5-21-1417001333-4928...	IUSR_SAT2-DC	Internet Guest Account	Built-in account for anon...	g	1	1
6	S-1-5-21-1417001333-4928...	IWAM_SAT2-DC	Launch IIS Process Account	Built-in account for Inte...	g	1	1
7	S-1-5-21-1417001333-4928...	TestUser	Test User		g	1	1
8	S-1-5-21-1482476501-1993...	Administrator		Vordefiniertes Konto für ...	g	2	1
9	S-1-5-21-1482476501-1993...	Gast		Vordefiniertes Konto für ...	g	2	1

C333 (8.0) C333\Administrator (55) sat 9 rows Ln 1, Col 1

Abbildung 3.7: Tabelle „users“

Abbildung 3.7 zeigt die Tabelle „users“ der *SAT2* Datenbank, in der alle relevanten Informationen über Benutzer bzw. Benutzerkonten gespeichert werden. Neben dem SID wird sowohl der Name des Benutzerkontos und der vollständige Name des Benutzers, als auch der Kommentar, ein Flag zur Unterscheidung zwischen lokalen und globalen Benutzerkonten, die Computer ID, der Maschine, auf der das Konto ausgelesen wurde und zuletzt die Versionsnummer des *Security-Scans*, gespeichert.

Im Gegensatz zu *Built-in Groups* können Informationen über *Built-in User*, wie z.B. Administrator oder Gast, sehr wohl vom *CGU-Scanner* ausgelesen werden. *Built-in*

User werden zwar auch durch gebräuchliche Sicherheitskennungen identifiziert, die in diesem Fall jedoch nicht konstant sind, sondern immer mit dem SID des Computers, auf dem ein Benutzerkonto lokalisiert wird, kombiniert werden.

In Hinblick auf spätere Auswertungen spielen auch diese Daten eine wesentliche Rolle, da zumeist die Berechtigungen bestimmter Benutzer analysiert werden. Unter Verwendung des Benutzer SIDs kann festgestellt werden, auf welche Objekte im Netzwerk der Benutzer Zugriffsrechte hat.

3.6.1.4 Gruppenmitgliedschaften (Membership)

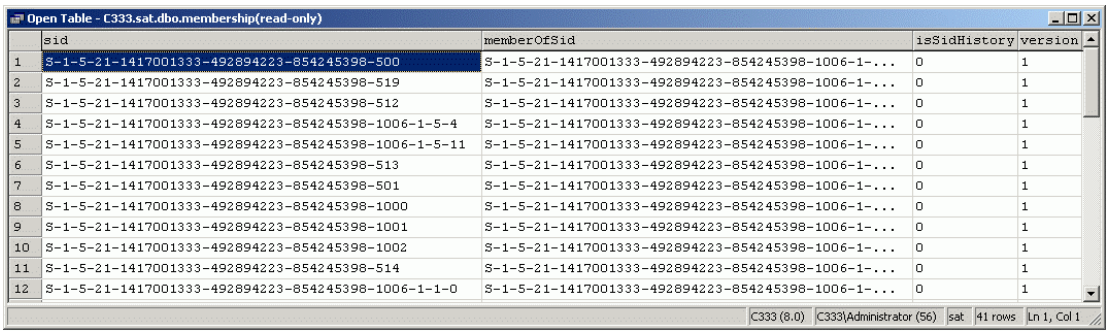
Wie aus der Beschreibung des Benutzer- u. Gruppenkonzeptes hervorgeht, können Benutzergruppen dazu verwendet werden, Benutzer mit gleichen Eigenschaften in Bezug auf deren Administration, zu Gruppen zusammenzufassen. Dies vereinfacht nicht nur die Vergabe der Berechtigungen, sondern ermöglicht auch eine übersichtliche und einfache Strukturierung der Benutzer in einem Netzwerk.

Ein Benutzer kann seine Zugriffsrechte also auf zwei unterschiedliche Arten erhalten. Einerseits „ad personam“, d.h. der Administrator legt explizit fest, mit welchen Rechten der Benutzer auf bestimmte Objekte zugreifen darf und andererseits durch seine Gruppenzugehörigkeiten, d.h. der Benutzer erhält durch die Mitgliedschaft in verschiedenen Gruppen zusätzliche Berechtigungen.

Aus diesem Grund muss bei späteren Auswertungen der Benutzerberechtigungen also auch die Gruppenmitgliedschaft berücksichtigt werden, um beispielsweise die „effektiven“ Rechte eines Benutzers ermitteln zu können (siehe Kapitel 3.7.2).

Zu diesem Zweck werden vom *CGU-Scanner* sämtliche Gruppenmitgliedschaften auf den zu analysierenden Maschinen ausgelesen und in der *SAT2* Datenbank geschrieben. In einer speziell dafür vorgesehenen Tabelle („membership“) werden ausschließlich die SIDs der Mitglieder („sid“) und der Gruppen, denen sie angehören („memberOfSid“) gespeichert. Nähere Informationen über die Teilhaber einer Mitgliedschaft müssen den Tabellen „users“ bzw. „groups“ durch Abfragen über den SID entnommen werden.

Abbildung 3.8 zeigt die Tabelle „membership“ nach einem erfolgreich durchgeführten *Security-Scan*.


Open Table - C333.sat.dbo.membership(read-only)

	sid	memberOfSid	isSidHistory	version
1	S-1-5-21-1417001333-492894223-854245398-500	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
2	S-1-5-21-1417001333-492894223-854245398-519	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
3	S-1-5-21-1417001333-492894223-854245398-512	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
4	S-1-5-21-1417001333-492894223-854245398-1006-1-5-4	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
5	S-1-5-21-1417001333-492894223-854245398-1006-1-5-11	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
6	S-1-5-21-1417001333-492894223-854245398-513	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
7	S-1-5-21-1417001333-492894223-854245398-501	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
8	S-1-5-21-1417001333-492894223-854245398-1000	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
9	S-1-5-21-1417001333-492894223-854245398-1001	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
10	S-1-5-21-1417001333-492894223-854245398-1002	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
11	S-1-5-21-1417001333-492894223-854245398-514	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1
12	S-1-5-21-1417001333-492894223-854245398-1006-1-1-0	S-1-5-21-1417001333-492894223-854245398-1006-1-...	0	1

C333 (8.0) | C333\Administrator (56) | sat | 41 rows | Ln 1, Col 1

Abbildung 3.8: Tabelle „membership“

Ein wesentlicher Punkt ist auch das Speichern der sogenannten *SID-History*, die ebenfalls bei der Analyse von Benutzerberechtigungen berücksichtigt werden muss.
Mit dem Erscheinen von Windows 2000 hat sich das Format der bisher verwendeten *Security Identifier* (SIDs) geändert. Da aber die Möglichkeit besteht, beispielsweise einen Windows NT Server in ein Windows 2000 Netzwerk zu integrieren, musste Microsoft eine Lösung, für das aus der Änderung der SIDs resultierende Problem finden.

Dieses Problem lässt sich folgendermaßen schildern: Würde ein Benutzerkonto einer NT Domäne ins *Active Directory* transferiert werden, hat dies eine Änderung des SIDs, der den Benutzer eindeutig im System identifiziert, zur Folge. Dadurch würde der Benutzer jedoch auf einen Schlag all seine bisherigen Berechtigungen verlieren. Um die Kompatibilität zwischen den unterschiedlichen SIDs zu gewährleisten, führte Microsoft die *SID-History* ein, in der alle SIDs, welche ein Sicherheitsprincipal hatte, gespeichert werden. Damit liegt in der *SID-History* in diesem Fall der „alte“ SID eines Benutzers, welchen er im *NT4 Domain* hatte.

Um diese SIDs in späteren Auswertungen berücksichtigen zu können, wird die *SID-History* im *Active Directory* vom *ADS-Scanner* ausgelesen und in der „membership“-Tabelle gespeichert. Gekennzeichnet werden SIDs aus der *SID-History* durch das Attribut „isSidHistory“ (siehe dazu [Hel/S.112 f.]). Dies ist notwendig, da für die Berechtigungen, welche der Benutzer „ad personam“ erhält, die SIDs aus der *SID-History* im-

mer mit einzubeziehen sind. Die anderen „normalen“ Gruppenmitgliedschaften jedoch sind nur bei der Berechnung der „effektiven Rechte“ zu berücksichtigen.

3.6.2 Der ADS-Scanner

Der *Active Directory Security – Scanner* stellt eine weitere Datensammler-Komponente des Sicherheitsanalyse-Tools dar. Der *ADS-Scanner* ermittelt die sicherheitsrelevanten Informationen aus dem *Active Directory*. Der *ADS-Scanner* startet erst nach erfolgreichem Abschluss der Benutzer- und Gruppenanalyse durch den *CGU-Scanner*, da gewisse Daten aus diesem Bereich vom *ADS-Scanner* verwendet werden (z.B. die Computer ID aus der Datenbank zum Lokalisieren von Objekten).

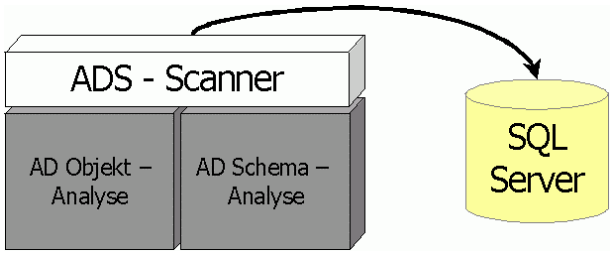


Abbildung 3.9: Architektur ADS-Scanner

Obgleich der *ADS-Scanner* von Hrn. Helml [Hel] entwickelt wurde, soll diese Komponenten hier doch genauer erläutert werden, da sich die Auswertung stark auf die *Active Directory* Daten konzentriert. Diese Entscheidung wurde getroffen, da das *SAT2* Team der Ansicht war (und noch immer ist), dass die Auswertungen von *NTFS* und *Registry* „nur“ vereinfachte Subsets der AD-Auswertung darstellen.

Der *ADS-Scanner* selbst lässt sich wiederum in zwei Komponenten unterteilen, wie aus Abbildung 3.9 ersichtlich ist. Eine der beiden Komponenten ist für die Analyse von Objekten im *Active Directory* zuständig, die andere für eine Analyse des *Active Directory Schemas*.

Folgende Analysen können unabhängig voneinander, und bei Bedarf auch auf unterschiedlichen Maschinen, durchgeführt werden:

- ***Objekt-Analyse:***

  Die Sicherheitseinstellungen (i.d.R. Zugriffssteuerungslisten) aller Objekte im *Active Directory* werden analysiert und in den entsprechenden Tabellen der *SAT2* Datenbank gespeichert. Zusätzlich werden zur Auswertung allgemeine Informationen über das Objekt benötigt, wie z.B. Name, Objekttyp, etc., und deshalb ebenfalls gespeichert.

- ***Schema-Analyse:***

  Durch das *Active Directory Schema* wird festgelegt, wie Objekte im Verzeichnis „auszusehen" und welchen Regeln sie zu gehorchen haben. Diese Beschreibung der Daten wird im Zuge der *Schema*-Analyse in der *SAT2* Datenbank gespeichert.

Eine Aufteilung des *ADS-Scanners* in diese beiden von einander unabhängigen Komponenten, war auf Grund der Natur des *Active Directory* erforderlich. Das Verzeichnis selbst kann auf mehrere *Domain-Controller* in verschiedenen Domains verteilt werden. Dabei wird auf allen DCs innerhalb einer Domain (durch Replikation) die gleiche Information gespeichert. Die Objekt-Analyse muss deshalb pro Domain auf je einem DC durchgeführt werden. Das Schema hingegen wird auf jeden *Domain-Controller* im gesamten *Active Directory* repliziert und enthält daher immer die gleichen Informationen. Demnach ist eine *Schema*-Analyse nur auf insgesamt einem DC innerhalb eines AD erforderlich.

Durch diese Gliederung des *ADS-Scanners* in die beiden o.a. Komponenten kann das *Active Directory* effizient analysiert werden. Dabei kann der Anwender selbst bestimmen, welcher Teil des *Active Directory* auf welchem *Domain-Controller* gescannt werden soll.

Um während einer *Active Directory* Analyse den SQL-Server zu entlasten, wurden in den *Scanner* spezielle *Caching-Routinen* integriert. Mittels einer *Move-To-Front* Liste werden die verschiedenen Zugriffssteuerungslisten (ACLs) lokal im Hauptspeicher gehalten. Bedenkt man, dass alleine bei einer Basis-Installation des *Active Directory* ~2600 Objekte analysiert werden und dafür lediglich ~60 verschiedene ACLs existieren,

für jedes Objekt aber geprüft werden muss, ob bereits eine entsprechende ACL gespeichert wurde, so können dadurch die Datenbankzugriffe ca. um den Faktor 40 reduziert werden. Da Datenbankzugriffe die Geschwindigkeit der Datensammler-Komponenten maßgeblich – im negativen Sinn – beeinflussen, erreicht man durch das *Caching* der ACLs eine beträchtliche Steigerung der Performance.

Eine Möglichkeit, um die letztendlich auszuwertende Datenmenge reduzieren zu können, bietet die Komprimierung von Objekten, welche optional (durch entsprechende *Scanner*-Konfiguration in der *Registry*) verwendet werden kann. Diese bezieht sich vorwiegend auf das Zusammenfassen jener AD-Objekte, die gleiche Eigenschaften aufweisen. Ein weiterer Komprimierungsschritt wurde unter der Annahme, dass für einen Administrator im Zuge einer Sicherheitsanalyse nur Abweichungen vom Regelfall von Bedeutung sind, implementiert.

Neben einer vollständigen Analyse des *Active Directory* (Komprimierungsstufe: 0), stellt der *ADS-Scanner* folgende Komprimierungsstufen zur Verfügung:

- ***Stufe 1: Zusammenfassen von gleichen Objekten***
  Gleichen Objekten, d.h. Objekte mit gleichem Objekttyp und gleicher ACL auf der selben Hierarchie-Ebene im Verzeichnisbaum, werden zusammengefasst und in Form eines repräsentativen Objektes in der *SAT2* Datenbank gespeichert. Zusätzlich wird die Anzahl der zusammengefassten Objekte mitgespeichert.

- ***Stufe 2: Abweichungen vom Regelfall – nur Ausnahmen***
  Es werden nur jene Objekte gespeichert, welche die Vererbung unterbrechen und/oder Abweichungen gegenüber der Standard-Security aus dem Schema aufweisen

- ***Stufe 3: Kombination***
  Sequentielle Ausführung der beiden ersten Komprimierungsstufen.

HINWEIS: Detaillierte Informationen über das *Active Directory* und den *ADS-Scanner* können der Diplomarbeit von Hrn. Helml [Hel] entnommen werden.

3.6.3 Der NTFS/REG-Scanner

Eigentlich stellen der *NTFS*- und *REG-Scanner* zwei voneinander unabhängige Teile der Datensammler-Komponente im *SAT2* System, dar. Zu Dokumentationszwecken werden sie in dieser Diplomarbeit jedoch nicht getrennt behandelt.

Das Grundprinzip des *NTFS/REG-Scanners* ist gewissermaßen ähnlich zu dem des *ADS-Scanners*. In Abhängigkeit von der *Scanner*-Konfiguration werden Sicherheitseinstellungen und Informationen über Objekte aus einem speziellen sicherheitsrelevanten Bereich des Netzwerks – in diesem Fall *NTFS* und *Registry* – gesammelt und direkt in die Datenbank geschrieben. Diese Daten stellen wiederum die Basis für spätere Analysen der Benutzerberechtigungen in den angesprochenen Bereichen des Netzwerks dar.

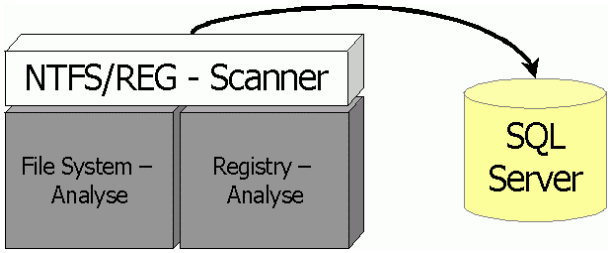


Abbildung 3.10: Architektur NTFS/REG-Scanner

Trotz der wesentlich geringeren Komplexität des Dateisystems und der Windows Registrierungsdatenbank im Vergleich zum *Active Directory*, sind Analysen in diesem Bereich nicht weniger aufwändig. Auch in diesem Fall sind Zugriffssteuerungslisten (ACLs) und Vererbungsmechanismen zu berücksichtigen.

Im Gegensatz zum *Active Directory* existieren im *NTFS* aber bei weitem weniger Berechtigungen, die für Objekte vergeben werden können, und lediglich zwei verschiedene Objekttypen: Ordner (*Container/Folder*) und Dateien (*Files*). Diese Tatsache vereinfacht sowohl die Analyse des Dateisystems als auch spätere Auswertungen, da nicht unzählige verschiedene Zugriffsrechte und Objekttypen berücksichtigt werden müssen.

Die *Windows-Registry* kann in Bezug auf Sicherheitseinstellungen im Vergleich zu *Active Directory* und *NTFS* nochmals als Vereinfachung angesehen werden. Die Zugriffsrechte beschränken sich in diesem Fall lediglich auf *Container*, sind im Verhältnis zum

NTFS noch weniger an der Zahl, und auch der Vererbungsmechanismus lässt weniger Variationen zu.

Auf Grund der zu erwartenden hohen Datenmenge können mit Hilfe des Controller-Programms, Start- und Endpunkte für Analysen des Dateisystems spezifiziert werden. Darüber hinaus wurden auch im *NTFS/REG-Scanner* die in Kapitel 3.6.2 bereits angesprochenen ACL *Caching-Routinen* und diverse Komprimierungsstufen implementiert. Ein durchaus realistischer Wert für die Anzahl der zu analysierenden Objekte im Dateisystem (bei Komprimierungsstufe 0) beläuft sich auf ~100.000+ Objekte.

Deshalb stehen folgende Komprimierungsstufen, die durch entsprechende Konfiguration des *NTFS/REG-Scanners* verwendet werden können, zur Verfügung:

- ***Stufe 1: Zusammenfassen von gleichen Objekte***
  Objekte gleichen Typs und mit gleicher ACL werden zusammengefasst und in Form eines repräsentativen Objektes in Kombination mit der Anzahl der zusammengefassten Objekte in der *SAT2* Datenbank gespeichert.

- ***Stufe 2: Zusammenfassen von Objekten eines Verzeichnisses***
  Weisen alle Dateien (Objekte) in einem Verzeichnis die gleiche *Security* auf, werden sie zu *.* zusammengefasst und als ein Datensatz in die *SAT2* Datenbank geschrieben. Ansonsten ist die Funktionalität vergleichbar mit der von Stufe 1.

- ***Optionale Einstellungen:***
  In Kombination mit den beiden o.a. Komprimierungsstufen können nur Ausnahmen, d.h. Objekte, die Brüche in der Vererbung oder Einträge in der „normalerweise“ leeren Standard-Security aufweisen, gespeichert bzw. zusätzlich Dokumente gleicher Applikationen zusammengefasst werden.

HINWEIS: Detaillierte Informationen über das *NTFS 5*, die *Windows-Registry* und den *NTFS*- bzw. *REG-Scanner* können der Diplomarbeit von Hrn. Achleitner [Ach] entnommen werden.

3.7 Visualisierung: SAT2 MMC Snap-in

Das *SAT2 MMC Snap-in* stellt den Hauptteil der praktischen Arbeit des Autors dar. Deshalb soll die im Rahmen dieser Diplomarbeit entstandene Visualisierungs-Komponente des *Security Analysis Tools 2 (SAT2)* im folgenden Abschnitt ausführlich beschrieben werden. Neben einer Dokumentation und Anleitung zur Verwendung des *Snap-in* werden auch die Konzepte zur Analyse der Benutzerberechtigungen und zur Darstellung derselben erläutert.

An dieser Stelle soll ausdrücklich darauf hingewiesen werden, dass es sich bei dem hier beschriebenen *SAT2 MMC Snap-in* lediglich um einen *Prototyp* handelt und nicht um die vollständige Implementierung einer Applikation zur Sicherheitsanalyse. Diese Entscheidung wurde aus zwei Gründen getroffen:

1) Weder der Umfang noch die Komplexität, den die Implementierung der Visualisierungs-Komponente als *MMC Snap-in* und die Analyse des *Active Directory* mit sich brachten, waren – trotz der Erfahrungen aus dem *SAT1* Projekt – abzusehen.

2) Durch die zahlreichen Gespräche und Diskussionen zwischen dem Projektleiter Hrn. Hörmanseder und dem Autor dieser Arbeit, wurde ein wesentlicher Faktor immer deutlicher: es ist nahezu unmöglich *DIE* Applikation zur Sicherheitsanalyse zu entwickeln. Die Subjektivität, welche unterschiedliche Darstellungsarten, die Farbwahl zur Einfärbung der Knoten oder eine Aufbereitung der Zugriffsrechte in sich birgt, hat maßgeblichen Anteil daran. Wir gehen daher von einem „evolutionären“ Ansatz aus. Bestimmte Komprimierungsarten und Darstellungsformen werden realisiert und auf Basis der dabei gewonnenen Erkenntnisse werden dann bei Bedarf neue Varianten bzw. Darstellungen eingebaut.

Deshalb wurde ein Prototyp auf *Framework-Basis* entwickelt, der so konzipiert wurde, dass möglichst einfach neue Auswertungsformen und Abfragen hinzugefügt bzw. implementiert werden können.

Im Zuge dessen wurden auch „beispielhafte“ Auswertungsfunktionen zur Analyse der Benutzerberechtigungen im *Active Directory* implementiert, die einerseits Aufschlüsse über das zu Projektbeginn relative unbekannte *Active Directory* bringen und andererseits repräsentativ für die Funktionsweise des Prototyps sein sollten.

Folgender Abschnitt dieser Diplomarbeit soll also nicht nur die praktische Arbeit des Autors beschreiben, sondern sozusagen auch als Leitfaden für *SAT2* Anwender und Entwickler, die den Prototyp weiterentwickeln wollen, dienen.

Zuerst soll aber anhand einer Dokumentation vor allem der Prototyp an sich vorgestellt, sowie die verschiedenen Einstellungsmöglichkeiten und verfügbaren Abfragen (Analysen) erläutert werden. In den darauffolgenden Kapiteln werden dann ausführlich jene Konzepte beschrieben, welche die Basis für Auswertungen und deren Darstellung bilden.

3.7.1 Dokumentation

Das Ziel, welches vorrangig mit der Entwicklung des *SAT2 Snap-ins* verfolgt wurde, war dem Anwender (i.d.R. einem Administrator) eine *Benutzer-zentrierte* Auswertung der Benutzerberechtigungen zu ermöglichen. Zu diesem Zweck wird allerdings ausschließlich auf die Daten, die durch die verschiedenen *Security-Scanner* in der Datenbank gespeichert wurden, zurückgegriffen. Folglich können durch das *SAT2 Snap-in* Sicherheitslöcher oder kritische Punkte in einem Netzwerk nur aufgezeigt, jedoch keine Sicherheitseinstellungen korrigiert werden. Dazu müssen nach wie vor Windows Standard-Werkzeuge verwendet werden.

Eine Erweiterung in Richtung eines „integrierten“ Produktes wurde zwar angedacht, insbesondere aus Gründen des Gesamtumfanges der Diplomarbeit aber derzeit nicht implementiert.

Wesentlich ist auch die Tatsache, dass das *SAT2 Snap-in* speziell für *Administratoren* entwickelt wurde, die im Umgang mit dieser Materie vertraut sind. Dies zeigt sich beispielsweise in der Gestaltung der Benutzerschnittstelle und der Darstellung der Benut-

zerberechtigungen. Dabei wurde besonderer Wert auf eine prägnante Darstellung der Zugriffsrechte, sowie auf eine effiziente Handhabung des Tools und weniger auf die allgemeine Verständlichkeit für „Laien“ gelegt.

Ausgehende von der Problemstellung, der Projektspezifikation und all den bisher beschriebenen Faktoren lassen sich die Anforderungen an das *SAT2 Snap-in* folgendermaßen – in einer Art Aufgabenstellung – zusammenfassen:

- Alle Auswertungen werden ausschließlich auf Basis der Daten, welche durch die verschiedenen *Scanner* in der Datenbank gespeichert wurden, durchgeführt.

- Die Analyse der Gruppenmitgliedschaften aller Benutzer und Gruppen des gescannten Netzwerkes kann zwar auch mit Windows Standard-Werkzeugen erfolgen, soll aber dennoch vom *SAT2 Snap-in* unterstützt werden.

- Das Hauptaugenmerk liegt auf einer *Benutzer-zentrierten* Analyse der Zugriffsrechte am Beispiel *Active Directory*.

- Die übersichtliche Darstellung der Analyseergebnisse in Form verschiedener Darstellungsarten soll u.a. als Basis für künftige Weiterentwicklungen dienen.

- Die Struktur des analysierten Netzwerks soll so dargestellt werden, dass Domänen-, Computer-, Gruppen- und Benutzerzugehörigkeiten daraus erkennbar sind und somit das Erstellen neuer Abfragen erleichtern.

Das auf Basis dieser Aufgabenstellung entwickelte *SAT2 MMC Snap-in* lässt in drei Hauptkomponenten unterteilen, die auf den folgenden Seiten beschrieben werden sollen (siehe auch Abbildung 3.11).

- ***Query Viewer:***
  Bietet eine Übersicht über die vom Anwender erstellten Abfragen.

- ***Result Viewer:***

  Dient als Basis-Container zur Darstellung der Analyseergebnisse

- ***Structure Viewer:***

  Repräsentiert die Struktur des analysierten Netzwerks.

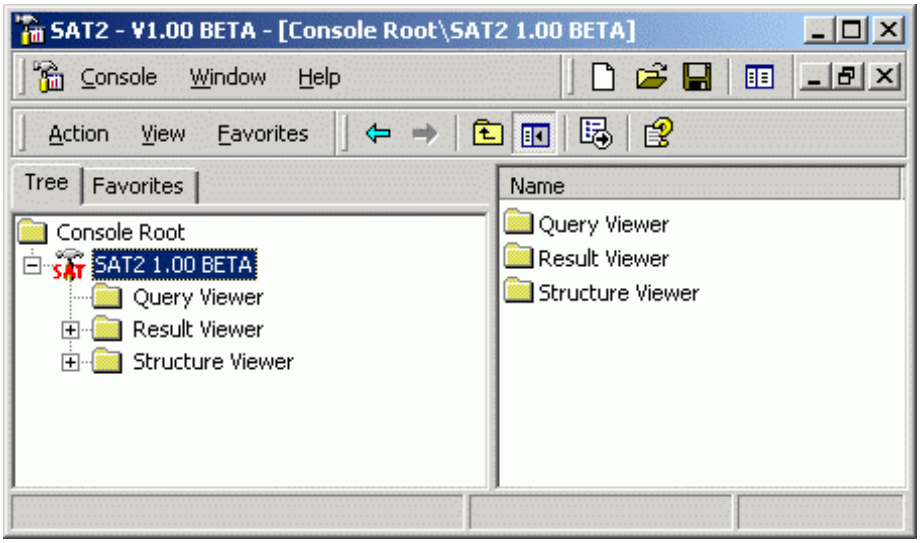


Abbildung 3.11: Komponenten des SAT2 Snap-ins

3.7.1.1 Der Query Viewer

Der erste Schritt zur Durchführung einer Analyse besteht darin, eine Abfrage (*Query*) zu definieren. Dabei kann vom Anwender selbst bestimmt werden, für welches *Security-Principal* er in welchem Bereich die Zugriffsrechte analysieren möchte. Der *Query Viewer* stellt jenen Container im *SAT2 Snap-in* dar, in welchem diese Abfragen gespeichert bzw. angezeigt werden und damit jederzeit an einem zentralen Ort zur Weiterverarbeitung oder näheren Betrachtung zur Verfügung stehen.

Zur Unterstützung des Anwenders beim Erstellen einer neuen Abfrage dient der „New Query“ – Dialog. Durch diesen Dialog können dem *Snap-in* die Grundeinstellungen für eine spätere Auswertungen übermittelt werden (siehe Abbildung 3.12).

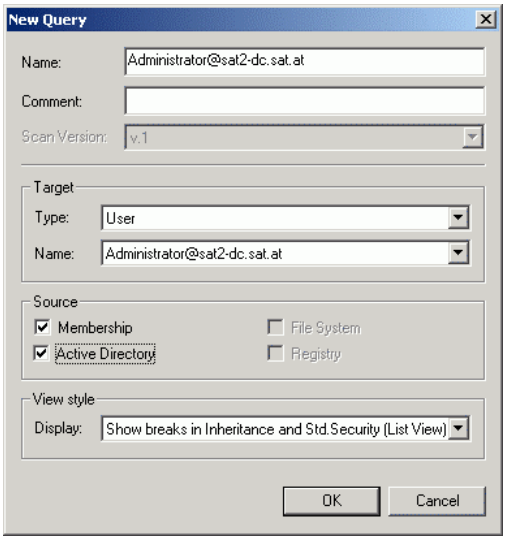


Abbildung 3.12: „New Query“ – Dialog

Neben dem Namen und einem optionalen Kommentar kann hier festgelegt werden, für welches *Security-Principal* (*Target*) die Auswertung durchgeführt, welcher sicherheitsrelevante Bereich analysiert und wie das Analyseergebnis dargestellt werden soll. In künftigen Implementierungen sollen Daten verschiedener *Security-Scans* ausgewertet werden können. Dazu wurde bereits jetzt eine Auswahlmöglichkeit für die Versionsnummer (*Scanversion*) des *Security-Scans* vorgesehen.

Die Auswahl des „Ziels“ erfolgt über die beiden Listenfelder im „Target“-Bereich. Dabei bezeichnet der Typ (*Type*) die Kategorie, welcher das *Security-Principal* zugeordnet werden kann. In dieser Version des *SAT2 Snap-ins* kann zwischen Computer (*Computer*), Gruppen (*Group*) oder Benutzern (*User*) ausgewählt werden. Abhängig vom selektierten Typ kann der Anwender im nächsten Schritt festlegen, für welches *Security-Principal* die Auswertung zu erfolgen hat. Dazu werden im zweiten Listenfeld (*Name*) alle *Security-Principals* des zuvor gewählten Typs aufgelistet, die durch den *CGU-Scanner* ermittelt werden konnten.

Anschließend muss die Quelle (*Source*), d.h. der zu analysierende sicherheitsrelevante Bereich des Netzwerks, spezifiziert werden. Die Auswahlmöglichkeiten beschränken sich in dieser Version des *SAT2 Snap-ins* auf die Analyse der Gruppenmitgliedschaften (*Membership*) und des *Active Directory*. *NTFS*- und *Registry*-Analysen sind erst in künftigen Implementierungen vorgesehen, deshalb sind die Kontrollkästchen zur Auswahl dieser beiden Bereiche momentan deaktiviert.

Zuletzt kann die Anzeigeart (*View Style*) bestimmt werden. In Abhängigkeit von dieser Einstellung werden dem Anwender die Analyseergebnisse auf unterschiedliche Weise

präsentiert. Die verschiedenen Darstellungsarten werden im Kapitel 3.7.5 ausführlich beschrieben und deshalb an dieser Stelle vernachlässigt.

Durch eine Bestätigung der Eingaben über den OK-Button werden diese an das *SAT2 Snap-in* weitergeleitet und anschließend sofort ausgewertet. Auf Basis dieser Informationen können in weiterer Folge alle Datenbankabfragen, die im Zuge einer Analyse durchzuführen sind, abgesetzt werden. Im selben Arbeitsschritt wird ein *Result Item* erzeugt, das diese Abfrage repräsentiert und im *Query Viewer* Container gespeichert.

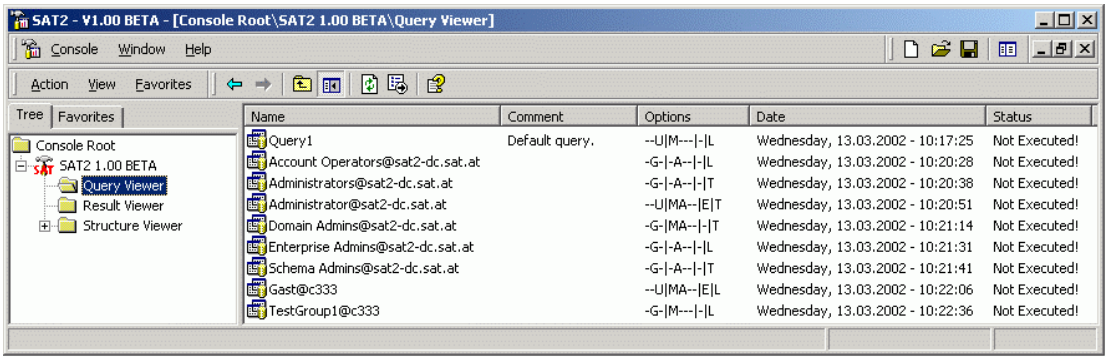


Abbildung 3.13: Query Viewer

Abbildung 3.13 zeigt den *Query Viewer*, der für die Anzeige der einzelnen Abfragen und ihrer wesentlichsten Eigenschaften zuständig ist.

Kategorie	Bezeichnung
Ziel:	C ... Computer
	G ... Group
	U ... User
Quelle:	M ... Membership
	A ... Active Directory
	F ... File System
	R ... Registry
Auswertung:	E ... Effective Rights
Darstellungsart:	L ... List View
	T ... Tree View

Tabelle 3.11: Legende zur komprimierten Darstellung der Optionen

Die Optionen, welche vom Anwender gewählt wurden, werden in komprimierter Form in der Spalte „Options“ angezeigt. Dies soll dem Anwender dahingehend helfen, die Unterschiede zwischen den einzelnen Abfragen auf einen Blick erkennen zu können –

um gewissermaßen den Überblick zu bewahren. Tabelle 3.11 gibt Aufschluss über die möglichen Buchstabenkombinationen, die für eine komprimierte Darstellung der Optionen in Frage kommen (vgl. dazu die Spalte „Options“ aus Abbildung 3.13).

Zur Überprüfung und Manipulation bereits erstellter Abfragen stehen dem Anwender im *Query Viewer* Eigenschaftenfenster (*Property Sheets*) zur Verfügung (siehe Abbildung 3.14). Diese Eigenschaftenfenster können u.a. auch dazu verwendet werden, bereits durchgeführte Abfragen entsprechend zu ändern und erneut auszuführen. Zusätzlich kann über die Erweiterten Einstellungen (*Advanced*) die Option zur Analyse der „effektiven“ Rechte eines Benutzers – unter Berücksichtigung seiner Gruppenmitgliedschaften und bestimmter „Well-known SIDs“ – ausgewählt werden. Standardeinstellung für alle Auswertungen ist die Analyse der *Trustee*-Rechte, d.h. jener Rechte, die der Benutzer „ad personam“ erhalten hat.

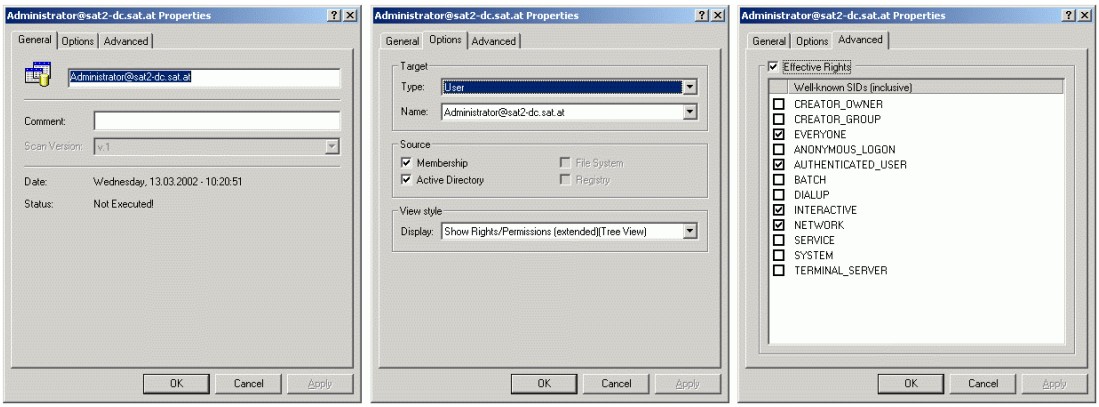


Abbildung 3.14: Eigenschaftenfenster im Query Viewer

Nachdem alle Einstellungen getätigt wurden, kann eine Analyse durchgeführt werden. Der Start einer Abfrage wird nach deren Selektion durch entsprechende Betätigung der Start-Schaltfläche in der Symbolleiste des *Snap-ins* oder durch Auswahl des Start-Befehls im Kontextmenü einer Abfrage initiiert.

3.7.1.2 Der Result Viewer

Der *Result Viewer* stellt das Kernstück des *SAT2 Snap-ins* dar. Er dient nicht nur als Basis-Container zur Anzeige der Analyseergebnisse, sondern birgt auch sämtliche Funktionalität, die zur Auswertung der Benutzerberechtigungen erforderlich ist, in sich.

Da im *Result Viewer* die Ergebnisse mehrerer verschiedener Analyseläufe gleichzeitig angezeigt werden können, wurde auf die Trennung und Strukturierung der einzelnen Ergebnisse besonderer Wert gelegt. Dies verläuft für alle Auswertungen nach folgendem Schema:

Nachdem eine Abfrage vom *Query Viewer* aus gestartet wurde, wird im *Result Viewer* zuerst ein Sohn-Knoten eingefügt, der wiederum als Wurzel für eine spezielle Abfrage angesehen werden kann und mit dem Namen dieser Abfrage bezeichnet ist. Eine Hierarchieebene darunter wird unter Berücksichtigung der Abfrageeigenschaften jeweils ein Ordner für den zu analysierenden sicherheitsrelevanten Bereich angelegt. Erst in der nächsten Ebene findet man die eigentlichen Informationen, die Aufschluss über die Zugriffsrechte geben sollen. Ab diesem Knoten wird die Struktur des zu analysierenden Bereichs in Abhängigkeit von der gewählten Darstellungsart und den Daten aus der Datenbank aufgebaut.

Zur Veranschaulichung der Struktur, die im Zuge einer Analyse im *Result Viewer* aufgebaut wird, soll Abbildung 3.15 dienen. Zu diesem Zweck wurde eine Analyse der Berechtigungen des Benutzers „Administrator“ im *Active Directory* durchgeführt.

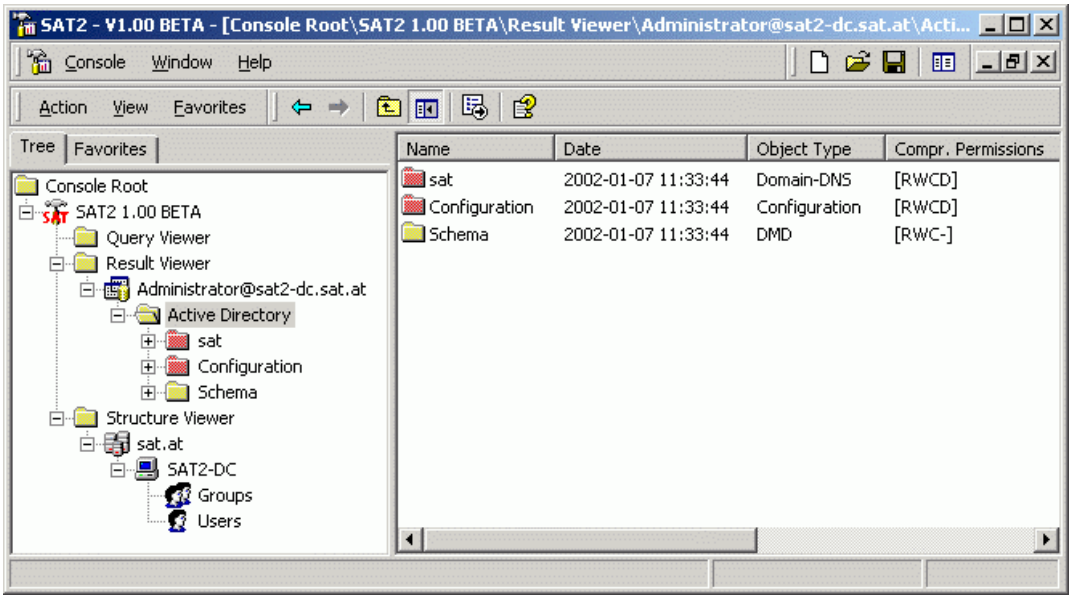


Abbildung 3.15: Result Viewer

Wie aus dieser Abbildung hervorgeht, werden in der Auswertung zum jeweiligen Objekt u.a. die Benutzerberechtigungen in der Spalte „Compr. Permissions“ angezeigt (siehe dazu Kapitel 3.7.4).

Zusätzlich können über Eigenschaftenfenster detaillierte Informationen der Berechtigungen anzeigt werden, die möglicherweise durch die komprimierte Darstellung verloren gegangen bzw. teilweise nicht einsehbar sind. Dazu wird die entsprechende Zugriffsteuerungsliste (ACL) des jeweiligen Objektes aufgeschlüsselt (siehe dazu auch Kapitel 3.7.4 und Kapitel 4).

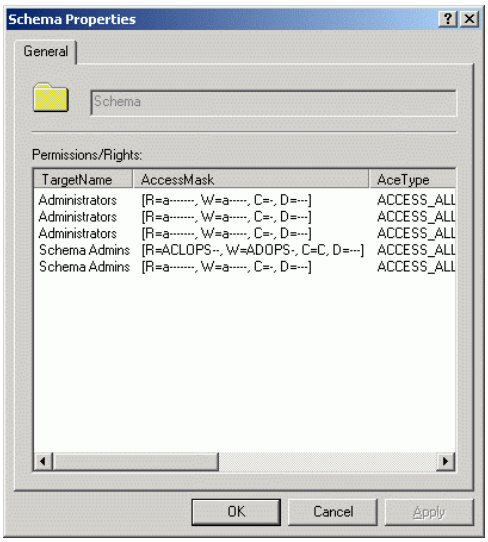


Abbildung 3.16: Auflistung der ACEs im Eigenschaftenfenster

Die genau Bedeutung einer Auswertung, wie sie in Abbildung 3.15 bzw. Abbildung 3.16 zu sehen ist, wird erst in der abschließenden Fallstudie und vor allem nach der Beschreibung der Grundkonzepte, welche für eine Auswertungen verwendet werden, erklärt.

3.7.1.3 Der Structure Viewer

Der *Structure Viewer* repräsentiert die Struktur des analysierten Netzwerkes. Dabei werden auf Basis der Daten der *SAT2* Datenbank die Zugehörigkeiten von Computern zu Domänen, sowie von Gruppen und Benutzern zu Computern, strukturiert dargestellt (siehe Abbildung 3.17).

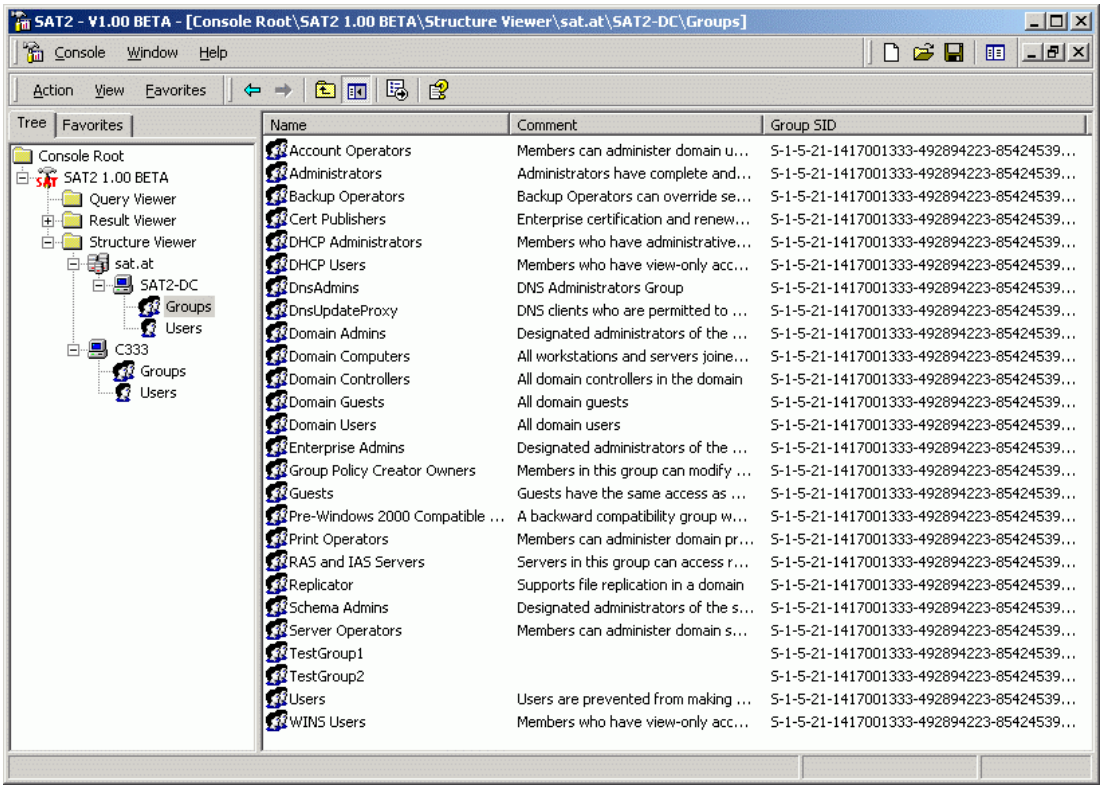


Abbildung 3.17: Structure Viewer

In erster Linie werden die Informationen über Domain-Namen aus der Tabelle „computers“ verwendet, um Container, welche die einzelnen Domänen repräsentieren sollen, zu erzeugen. In diesen Containern werden anschließend alle Computer zusammengefasst, die Mitglieder dieser Domäne sind. Zuletzt können durch gezielte Datenbankabfragen jene Gruppen und Benutzer ermittelt werden, die auf den verschiedenen Computern ausgelesen wurden. Diese werden den jeweiligen Computer-Knoten – in den dafür vorgesehenen Containern – zugeordnet.

Da prinzipiell die Möglichkeit besteht, Computer zu analysieren, die keiner Domäne angehören, muss auch für deren Anzeige gesorgt werden. Zu diesem Zweck werden „*stand-alone*“ Computer auf der selben Hierarchieebene, wie Domänen – als direkte Söhne des *Structure Viewer* Knotens – eingefügt.

Im Zuge der Implementierung des *SAT2 Snap-in* und der zahlreichen, zu Testzwecken durchgeführten Auswertungen, wurde man bald auch auf einige Schwächen der Benutzerschnittstelle aufmerksam, die beim ersten Design außer Acht gelassen wurden.

Das Konfigurieren einer Abfrage gestaltete sich bei zunehmender Zahl von *Security-Principals* immer schwieriger, da das zur Auswahl des „Ziel-Objektes“ vorgesehene Listenfeld im „New Query“ – Dialog, den Anwender leicht den Überblick verlieren lässt.

Die Frage, die sich in diesem Zusammenhang aufdrängte, war: warum nicht den *Structure Viewer* als Basis für das Konfigurieren neuer Abfragen zu verwenden. Aus der gegebenen Struktur können *Security-Principals* bei weitem einfacher ausgewählt werden, als aus dem zuvor angesprochenen Listenfeld. Darüber hinaus kann der Anwender auf diesem Weg leichter nachvollziehen, wo das *Security-Principal* im Netzwerk einzuordnen ist.

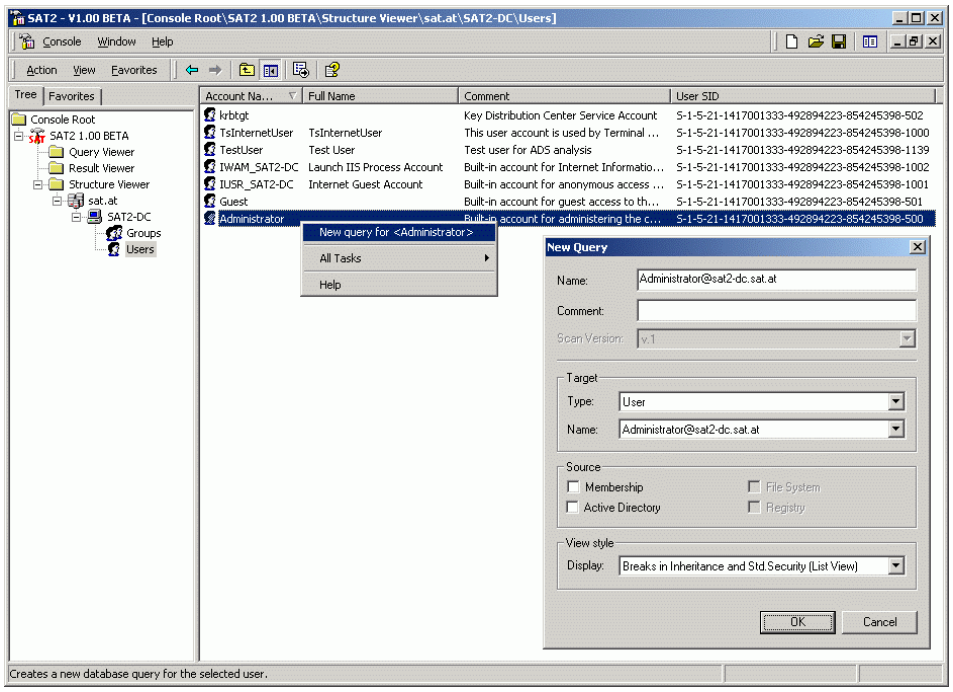


Abbildung 3.18: Erstellen einer Abfrage im Structure Viewer

Aus diesem Grund kann der in Kapitel 3.7.1.1 beschriebene „New Query“ – Dialog auch aus dem *Structure Viewer*, über das mit dem Objekt assoziierte Kontext-Menü, aufgerufen werden (siehe Abbildung 3.18).

3.7.2 Bestimmen der Gruppenmitgliedschaften

Wie bereits erwähnt bietet Windows NT/2000 (wie faktisch alle anderen Betriebssysteme auch) die Möglichkeit, Benutzer, welche die selben Eigenschaften aufweisen oder beispielsweise Zugriff auf die selben Ressourcen im Netzwerk benötigen, zu Gruppen zusammenzufassen. Auf diesem Weg können für eine Menge von Benutzern in einem Arbeitsschritt die gleichen Berechtigungen vergeben werden. Dabei wird – in der Regel vom Administrator – zuerst eine Gruppe erstellt und mit den erforderlichen Berechtigungen ausgestattet. Durch das Hinzufügen von Benutzern zu dieser Gruppe werden die entsprechenden Zugriffsrechte auf die (neuen) Mitglieder übertragen, d.h. jedes Mitglied übernimmt die Rechte der ihm zugewiesenen Gruppe(n). Benutzer, die zuvor schon Mitglied in einer (oder mehreren) anderen Gruppe(n) waren, verlieren dadurch ihre bisherigen Berechtigungen im Regelfall nicht, sondern erhalten durch diese neue Mitgliedschaft zusätzliche Rechte.

Dieses Gruppenkonzept muss aus zwei Gründen auch im *SAT2* System berücksichtigt werden:

1) ***Zur Analyse der Gruppenmitgliedschaften:***
   Für jeden Benutzer im Netzwerk muss seine Gruppenzugehörigkeit entsprechend dargestellt werden können.

2) ***Zur Analyse der Benutzerberechtigungen:***
   Jeder Benutzer kann durch seine Gruppenzugehörigkeit zusätzliche Berechtigungen erhalten, die Einfluss auf die Gesamtsumme seiner Zugriffsrechte haben können (effektive Rechte).

Die Mitglieder einer Gruppe repräsentieren in der Regel verschiedene Benutzer eines Netzwerkes. Die Flexibilität des Gruppenkonzeptes erlaubt jedoch auch Konstellationen in der Art, dass nicht nur Benutzer, sondern auch Gruppen als Mitglieder anderer Gruppen aufgenommen werden können (siehe Abbildung 3.19). Zusätzlich können Gruppen flexibel ineinander geschachtelt werden und beliebig viele Mitglieder haben bzw. selbst Mitglied in variabel vielen Gruppen sein.

Daraus resultieren die im Bezug als direkte und indirekte Mitgliedschaften bezeichneten Gruppenzugehörigkeiten, die in den Auswertungen der effektiven Rechte eines Benutzers unbedingt berücksichtigt werden müssen.

Im *SAT2* System können Auswertungen allerdings nicht nur auf Benutzer- sondern auch auf Gruppenebene durchgeführt werden. D.h. für den Anwender besteht zusätzlich zur Analyse von Benutzerberechtigungen die Möglichkeit, die Zugriffsrechte einer bestimmten Benutzergruppe auszuwerten. Um aber auch die effektiven Rechte einer Gruppe bestimmen zu können, müssen in diesem Fall natürlich alle direkten und indirekten Mitgliedschaften der zu analysierenden Gruppe berücksichtigt werden.

Durch Abbildung 3.19 soll der Einfluss des Gruppenkonzeptes auf Auswertungen mit dem *SAT2* System demonstriert werden. Zu diesem Zweck wurde ein einfaches Beispiel konstruiert, das die Mitgliedschaften eines Benutzers und dreier Gruppen darstellt. Die in der Abbildung eingezeichneten Pfeile markieren jeweils eine direkte Mitgliedschaft.

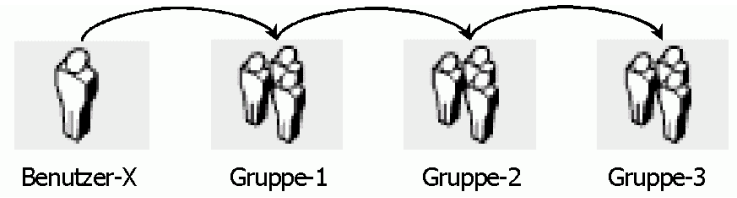


Abbildung 3.19: Gruppenmitgliedschaft

Im Falle einer Analyse der Gruppenzugehörigkeit oder der effektiven Rechte des Benutzers X muss sowohl die direkte Mitgliedschaft in Gruppe 1 berücksichtig werden, als auch die indirekten Mitgliedschaften in den Gruppen 2 und 3, obwohl Benutzer X offensichtlich kein (direktes) Mitglied dieser beiden Gruppen ist.

Aus dieser Grafik und den vorangegangenen Überlegungen lässt sich eine Tabelle (siehe Tabelle 3.12) aufstellen, die mehr Aufschluss über die Beziehungen zwischen dem Benutzer und den einzelnen Gruppen gibt. Dabei wird deutlich, wer Mitglied in welcher Gruppe ist und welche Gruppenzugehörigkeiten in einer Auswertung berücksichtigt werden müssen.

Security-Principal	Mitglied	Mitglied von	Auswertung
Benutzer-X	--	Gruppe-1	Gruppe-1 (d), Gruppe-2 (i), Gruppe-3 (i)
Gruppe-1	Benutzer-X	Gruppe-2	Gruppe-2 (d), Gruppe-3 (i)
Gruppe-2	Gruppe-1	Gruppe-3	Gruppe-3 (d)
Gruppe-3	Gruppe-2	--	--

Tabelle 3.12: Gruppenmitgliedschaften

In Hinblick auf die Implementierung lässt sich anhand dieser Tabelle leicht ein Algorithmus entwerfen, mit welchem die Gruppenzugehörigkeiten eines *Security-Principals*, das nicht zwangsläufig ein Benutzer sein muss, ermittelt werden können.

Aus praktischer Sicht lässt sich der Ablauf dieses Algorithmus folgendermaßen formulieren: Unter Verwendung des SIDs des Ziel-Objektes werden aus der Tabelle „membership“ alle direkten Mitgliedschaften des *Security-Principals* abgefragt. Als Zwischenergebnis erhält man in der Regel eine Menge von SIDs. Jeder dieser SIDs, der eine Gruppe identifiziert (Abfrage der Tabelle „groups“), ist für die zu bestimmenden Mitgliedschaften relevant und wird deshalb gespeichert. Für jeden SID der gespeichert wurde (d.h. es handelt sich um den SID einer Gruppe), müssen wiederum alle direkten Mitgliedschaften ermittelt werden. Erhält man abermals einen oder mehrere Gruppen-SIDs, müssen dafür wieder alle Mitgliedschaften ermittelt werden. Dieser Vorgang wird solange fortgesetzt, bis alle direkten und indirekten Gruppenmitgliedschaften bestimmt werden konnten.

Auf diese Weise kann man für jeden Benutzer und jede Gruppe, die durch den *CGU-Scanner* im System ausgelesen wurden, die Gruppenzugehörigkeiten feststellen und erhält dadurch eine Ausgangsbasis zur Darstellung der Mitgliedschaften bzw. zur Analyse der effektiven Rechte eines *Security-Principals*.

Zusätzlich ist hier auch die in Kapitel 3.6.1.4 erläuterte *SID-History* zu berücksichtigen. Wie bereits erwähnt müssen die SIDs aus der *SID-History* – um korrekte Analyse-Ergebnisse zu erhalten – bei allen Auswertungen mit einbezogen werden. Die „normalen“ Gruppenmitgliedschaften jedoch haben nur Einfluss auf die Berechnung der effektiven Rechte.

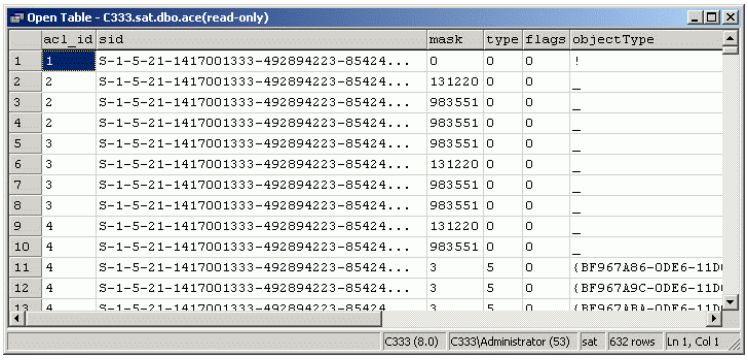
3.7.3 Analyse der Zugriffsrechte

Die Vergabe von Berechtigungen verläuft in Windows 2000 Systemen *Objekt-zentriert*. Dabei wird für jedes Objekt festgelegt, welches *Security-Principal* mit welchen Berechtigungen auf das Objekt zugreifen darf.

Diese Berechtigungen werden in Form sogenannter Zugriffssteuerungslisten (*Access Control Lists*, ACLs) zum Objekt gespeichert. Jede ACL setzt sich wiederum aus einem oder mehreren Zugriffssteuerungseinträgen (*Access Control Entries*, ACEs) zusammen, welche die Schutzmechanismen bestimmen, die für ein Objekt und seine Eigenschaften gelten. Ein ACE muss demnach zumindest folgende Informationen enthalten:

- Einen SID, der einen Benutzer oder eine Gruppe identifiziert
- Eine Bitmaske (*Access Mask*), durch die die Zugriffsrechte festgelegt werden
- Ein Flag, das den ACE Typ angibt
- Verschiedene Flags zur Festlegung der Vererbungsstrategie bestimmen

Betrachtet man in diesem Zusammenhang die *SAT2* Datenbank, kann man erkennen, dass in der Tabelle „ace“ genau diese Informationen durch die *Security-Scanner* gespeichert werden. Somit stehen alle wesentlichen Daten zur Analyse der Zugriffsrechte zur Verfügung.


Open Table - C333.sat.dbo.ace(read-only)

	acl_id	sid	mask	type	flags	objectType
1	1	S-1-5-21-1417001333-492894223-85424...	0	0	0	!
2	2	S-1-5-21-1417001333-492894223-85424...	131220	0	0	_
3	2	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
4	2	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
5	3	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
6	3	S-1-5-21-1417001333-492894223-85424...	131220	0	0	_
7	3	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
8	3	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
9	4	S-1-5-21-1417001333-492894223-85424...	131220	0	0	_
10	4	S-1-5-21-1417001333-492894223-85424...	983551	0	0	_
11	4	S-1-5-21-1417001333-492894223-85424...	3	5	0	{BF967A86-0DE6-11D
12	4	S-1-5-21-1417001333-492894223-85424...	3	5	0	{BF967A9C-0DE6-11D
13	4	S-1-5-21-1417001333-492894223-85424	3	5	0	{BF967AB4-0DE6-11D

C333 (8.0) | C333\Administrator (53) | sat | 632 rows | Ln 1, Col 1

Abbildung 3.20: DB-Tabelle „ace“

Bevor jedoch die eigentliche Auswertung durchgeführt werden kann, muss typischerweise festgelegt werden, für welchen *Security-Principal* die Analyse der Zugriffsrechte erfolgen soll. Das im folgenden Abschnitt beschriebene Verfahren bezieht sich auf die

Auswertung der Berechtigungen eines Benutzers, kann aber natürlich auch zur Auswertung von Gruppenberechtigungen eingesetzt werden.

Das Ziel, welches mit der Analyse der Zugriffsrechte verfolgt wird, ist all jene Objekte bestimmen zu können, auf die ein bestimmter Benutzer zugreifen darf (Abbildung 3.21 soll diese Vorgehensweise graphisch darstellen). Daraus resultiert die geforderte *Benutzer-zentrierte* Sicht. Welche Zugriffsrechte der Benutzer im Speziellen auf ein Objekt hat, ist zu diesem Zeitpunkt der Sicherheitsanalyse irrelevant.

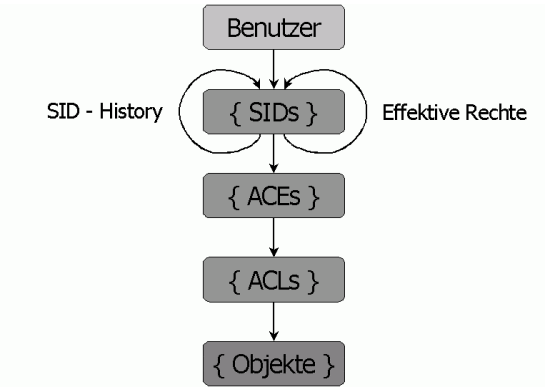


Abbildung 3.21: Analyse der Zugriffsrechte (Vorgehensweise)

Als erster Schritt ist der SID des Benutzers zu bestimmen. Dabei ist darauf zu achten, dass - in Abhängigkeit von der Konfiguration einer Abfrage - auch die effektiven Rechte eines Benutzers analysiert werden können und deshalb eine Menge von SIDs als Ausgangsbasis für die weitere Analyse zu verwenden ist. Das in Kapitel 3.7.2 beschriebene Verfahren zum Bestimmen der Gruppenmitgliedschaften kommt in diesem Fall auch hier zum Einsatz und ist in Abbildung 3.21 durch die beiden Schleifen (*SID-History* und effektive Rechte) gekennzeichnet.

Im zweiten Schritt wird auf Basis dieser SIDs die Menge aller ACEs bestimmt, die für diesen SID existieren. Eine gezielte Datenbankabfrage auf der „ace“-Tabelle über das Attribut „sid“ (siehe Abbildung 3.20) liefert das gewünschte Resultat. Zusätzlich erhält man durch diese Abfrage einen Verweis auf die jeweilige ACL, der ein ACE angehört (Attribut „acl_id“).

In einem letzten Schritt können durch die Menge der ACLs alle Objekte aus der Tabelle „objects“ abfragt werden, auf die der ausgewählte Benutzer Rechte hat.

Realisiert wird diese Vorgehensweise im *SAT2 Snap-in* durch relativ einfache SQL-Abfragen. Hier so nachmals darauf hingewiesen, dass alle Daten, die zur Sicherheitsanalyse benötigt werden, ausschließlich aus der *SAT2* Datenbank kommen. Die aus diesen Abfragen resultierende Menge von Objekten bildet den Grundstock jener Daten, die letztendlich im *Snap-in* angezeigt werden und dienen sozusagen als Basisdaten, die im Verlauf einer Sicherheitsanalyse weiterverarbeitet werden.

3.7.4 Komprimierung der Rechte

Nachdem nun alle für die Auswertung relevanten Objekte zur Verfügung stehen, müssen die jeweiligen Berechtigungen des ausgewählten Benutzers auf ein Objekt in übersichtlicher Form dargestellt werden.

Die Wahl einer „passenden“ Darstellungsart der Rechte gestaltete sich aus mehreren Gründen schwieriger als zunächst erwartet. Aber trotzdem war eine selbstgestellte Vorgabe, die Berechtigungen in ***einer*** Spalte im *Result Pane* des *Snap-in* anzuzeigen. Bei einer größeren Anzahl von Rechten bzw. Rechte-Kombinationen stellte sich heraus, dass die exakten Rechte mit dem vorhanden Platzangebot nicht übersichtlich darzustellen sind. Dazu kam weiters, dass die *Tooltips* im *Result Pane* nicht durch den Entwickler verändert werden können. Auch Listenfelder sind in der Listendarstellung nicht einsetzbar. So kann innerhalb einer Zeile der *Result Pane* z.B. kein *Drop-Down*-Menü angezeigt werden. Eine Anlehnung an die graphische Darstellung der Berechtigungen, wie sie in Windows Standard-Werkzeugen verwendet wird, d.h. die Anzeige der aufgeschlüsselten Rechte über Eigenschaftenfenster, schied ebenfalls aus.

Um dem Anwender (Administrator) die Möglichkeit zu bieten, auf einen Blick zu erkennen, welche Berechtigungen ein Benutzer auf einem Objekt hat, musste eine möglichst kompakte aber dennoch aussagekräftige Darstellungsart der einzelnen Rechte gefunden werden, die in Verbindung mit dem jeweiligen Objekt angezeigt werden kann.

Zu diesem Zweck werden die einzelnen Berechtigungen in komprimierter Form auf eine Zeichenkette abgebildet. Komprimierung in diesem Zusammenhang bedeutet aber lediglich ein Zusammenfassen bzw. Gruppieren von Berechtigungen, die ähnliche Auswirkungen haben, zu einer übergeordneten Gruppe oder das Abkürzen von Rechten durch entsprechende Buchstaben.

Da der *SAT2 Snap-in Prototyp* für Analysen im *Active Directory* konzipiert wurde, soll diese Komprimierung der Rechte auch am Beispiel des *Active Directory* erläutert werden.

Einem Objekt im *Active Directory* können 19 verschieden Rechte (*Access Rights*) zugewiesen werden. Diese werden als Bitmaske in der sogenannten (*Access Mask*) gespeichert. Zusätzlich können diese Rechte in Abhängigkeit vom Typ des ACE unterschiedliche Bedeutung haben, d.h. der ACE-Typ legt fest, ob sich ein Recht auf das gesamte Objekt oder nur auf gewisse Attribute des Objektes, bezieht (siehe dazu [Hel/S.53 ff.]). ***Anmerkung:*** Im Zuge der *Active Directory Analyse* werden vier dieser 19 Rechte, die sogenannten *Generic Rights*, vom *ADS-Scanner* in die verbleibenden 15 Rechte aufgelöst, was eine spätere Auswertung vereinfachen soll.

Die Vorgehensweise zur Komprimierung der Berechtigungen für ein zu analysierendes Objekt beginnt damit, dass alle den ausgewählten Benutzer betreffenden ACEs dieses Objektes aus der Datenbank abgefragt und zwischengespeichert werden. Die weitere Auswertung erfolgt anhand der Bitmaske (*Access Mask*) und des ACE Typs. Da sich die Berechtigungen für ein Objekt zumeist aus mehreren ACEs zusammensetzen, werden die einzelnen Bitmasken miteinander verknüpft.

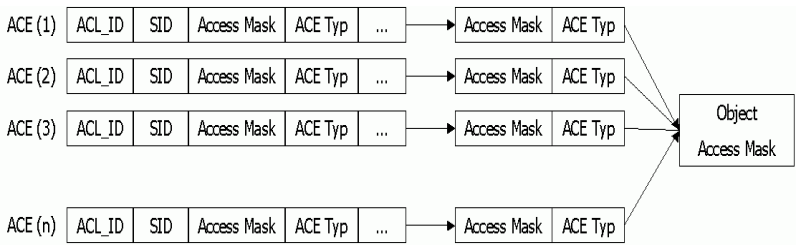


Abbildung 3.22: Zusammenfassen der Berechtigungen

Die daraus resultierende Bitmaske (*Object Access Mask*) bildet die Basis für alle im Verlauf einer Sicherheitsanalyse durchzuführenden Auswertungen, nicht nur zur Komprimierung der Rechte. Anhand dieser Bitmaske können beispielsweise auch einfache Vergleiche der Rechte verschiedener Objekte durchgeführt werden (siehe dazu Kapitel 3.7.6).

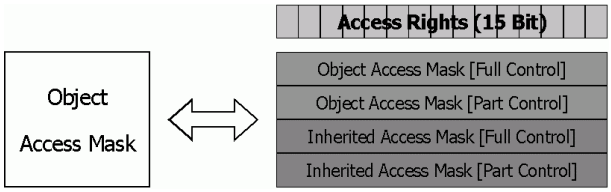


Abbildung 3.23: Object Access Mask (Implementierungs-Sicht)

Aus Implementierungs-Sicht ist es jedoch nicht empfehlenswert alle ACEs eines Objektes in einer Bitmaske zusammenzufassen. Wesentliche Informationen würden dabei verloren gehen und somit eine inkorrekte Darstellung der Berechtigungen nach sich ziehen. Aus diesem Grund wurde die *Object Access Mask* in 4 x 15 Bit *Access Masks* unterteilt (siehe Abbildung 3.23). In zwei dieser Bitmasken werden sämtliche Rechte gespeichert, welche das Objekt „per se“ erhalten hat (*Object Access Masks*). Eine davon enthält die Berechtigungen, die auf das „gesamte“ Objekt (*Full Control*), die andere, jene Berechtigungen, die nur auf gewisse Attribute (*Part Control*) vergeben wurden. Die als *Inherited Access Masks* bezeichneten Bitmasken enthalten alle vererbten Rechte, die wiederum in *Full* bzw. *Part Control* unterteilt wurden.

Ausgehend von nunmehr maximal 15 verschiedenen Rechten in je einer *Object Access Mask* wird im nächsten Schritt aus den beiden Bitmasken, welche die Objektberechtigungen enthalten, die gewünschte Zeichenkette generiert. Dazu wurde gemeinsam mit Hrn. Hörmanseder ein entsprechendes Muster als Beispiel entworfen, das eine Zuordnung der einzelnen Rechte zu einer von insgesamt vier Kategorien vorsieht (siehe Tabelle 3.15). In der letztendlich darzustellenden Zeichenkette werden die einzelnen Rechte also durch den Anfangsbuchstaben der Kategorie, welcher sie zugeordnet wurden, gekennzeichnet. Diese Buchstaben werden im Zuge der Auswertung aneinander gereiht und mit dem jeweiligen Objekt im *Snap-in* angezeigt. Analog dazu werden die vererbten Rechte auf eine zweite Zeichenkette abgebildet.

Aus dieser Vorgehensweise können sich beispielsweise folgende Darstellungsarten für die Rechte eines Benutzers auf einem Objekt ergeben. Hat der Benutzer alle notwendigen Rechte einer Klasse, so wird das Recht in Großschreibung dargestellt. Hat er dagegen nur einen Teil der Rechte, da wird die Kleinschreibung als Darstellung gewählt. Eine Liste der möglichen Rechte-Kombinationen dazu findet sich in Tabelle 3.15.

Kompr. Darstellung	Bedeutung
[----]	Keine Rechte
[RW--]	Lese- und Schreibrechte (Read [R] + Write [W])
[Rw--]	Lese- und Schreibrechte (Read [R] + Write nur teilweise, z.B. Lesen aller Eigenschaften eines User-Objektes, Schreibrechte nur auf einen Teil)
[--CD]	Rechte zum Löschen und Erzeugen eines Objektes (Create [C] + Delete [D])
[RWCD]	Max. Rechte (Full Control)

Tabelle 3.13: Komprimierte Darstellung von Rechten

Bisher noch nicht erwähnt wurde, dass im *Active Directory* Berechtigungen auch auf einzelne Attribute eines Objektes vergeben werden können. Diese Eigenschaft kann anhand des ACE Typs festgestellt werden. Der ACE Typ bestimmt also, ob die in der *Access Mask* festgelegten Berechtigungen auf das gesamte Objekt oder nur für Teile (Attribute) des Objekts beziehen. Diese Tatsache muss natürlich auch in der komprimierten Darstellung der Rechte berücksichtigt werden.

Dabei erweist sich der konsistente Einsatz der Groß- und Kleinschreibung als durchaus nützlich. Auf Basis der selben Rechte-Kategorien (Tabelle 3.15) werden Großbuchstaben verwendet, um zu kennzeichnen, dass alle jeweiligen Rechte für das gesamte Objekt gelten. Kleinbuchstaben zeigen nun Berechtigungen an, welche nur zum Teil oder nur auf bestimmte Teile des Objektes (Attribute) gelten. Die folgenden Beispiele sollen dies deutlich machen.

Kompr. Darstellung	Bedeutung
[----]	Keine Rechte
[Rw--]	Leserecht auf das gesamte Objekt, Schreibrecht nur auf einzelne Attribut(e)
[--cd]	Rechte zum Erzeugen und Löschen eingeschränkt auf bestimmte Objekte (als Beispiel siehe Abbildung 4.5)
[RWCD]	Max. Rechte (Full Control)

Tabelle 3.14: Rechtedarstellung unter Berücksichtigung des ACE Typs

Diese Form der komprimierten Darstellung von Berechtigungen wird im *SAT2 Snap-in* primär eingesetzt. Sie soll dem Anwender Aufschluss über den Sicherheitsstatus eines Objektes geben und wird in folgender Tabelle als Variante 1 bezeichnet. Tabelle 3.15 zeigt detailliert, wie die einzelnen Berechtigungen (*Access Rights*) des *Active Directory* der jeweiligen Kategorie zugeordnet und auf welche Buchstaben sie abgebildet werden.

Kategorie	Access Rights (ADS)	Variante 1	Variante 2
Read [R]	ADS_RIGHT_ACTRL_DS_LIST	R / r	L / l
	ADS_RIGHT_DS_LIST_OBJECT	R / r	O / o
	ADS_RIGHT_DS_READ_PROP	R / r	P / p
	ADS_RIGHT_READ_CONTROL	R / r	C / c
	ADS_RIGHT_SYNCHRONIZE	R / r	Y / y
Write [W]	ADS_RIGHT_DS_WRITE_PROP	W / w	P / p
	ADS_RIGHT_WRITE_DAC	W / w	D / d
	ADS_RIGHT_WRITE_OWNER	W / w	O / o
Create [C]	ADS_RIGHT_DS_CREATE_CHILD	C / c	C / c
Delete [D]	ADS_RIGHT_DELETE	D / d	O / o
	ADS_RIGHT_DS_DELETE_CHILD	D / d	C / c
	ADS_RIGHT_DS_DELETE_TREE	D / d	T / t
Read [R] + Write [W]	ADS_RIGHT_ACCESS_SYSTEM_SECURITY	R / r + W / w	T / t
	ADS_RIGHT_DS_CONTROL_ACCESS	R / r + W / w	A / a
	ADS_RIGHT_DS_SELF	R / r + W / w	S / s

Tabelle 3.15: Abbildung der Access Rights

Als Alternative zu der mit Variante 1 bezeichneten Darstellungsart wird dem Anwender die Möglichkeit geboten, jene Informationen über die einzelnen Berechtigungen einzusehen, die durch die zuvor beschriebene Komprimierung verloren gegangen sind. Dazu wird in einem Eigenschaftenfenster die komplette Zugriffssteuerungsliste (ACL) in Form aller ACEs, die auf den ausgewählten Benutzer zutreffen, angezeigt (siehe auch Abbildung 3.16). Um die jeweiligen Berechtigungen exakt darstellen zu können, wurde die in Tabelle 3.15 mit Variante 2 bezeichnete „komprimierte“ Darstellung entworfen. In diesem Fall werden die Rechte äquivalent zu Variante 1 gruppiert und wiederum in Abhängigkeit des ACE Typs entweder groß oder klein geschrieben. Die verschiedenen Buchstaben bezeichnen allerdings je ein ADS-Recht (Beispiel: [R=LOPCYTAS, W=PDOTAS, C=C, D=OCT]). Diese Darstellungsart wird in der Fallstudie (Kapitel 4) noch genauer beschrieben.

HINWEIS: Diese Form der Komprimierung und Darstellung der Rechte, wie sie für die Implementierung des *SAT2 Snap-ins* gewählt wurde, stellt lediglich ein Beispiel für die

zahlreichen unterschiedlichen Darstellungsmöglichkeiten dar. Durch die Flexibilität der Listenansicht im *Result Pane* des *Snap-ins* kann die jeweilige Darstellung der Berechtigungen vom Anwender selbst bestimmt werden. Darüber hinaus besteht für Entwickler die Möglichkeit neue Rechte-Darstellungen selbst zu entwerfen und mit nur geringem Aufwand in das *SAT2 Snap-in* zu integrieren.

3.7.5 Ausgewählte Darstellungsarten

Ähnlich wie die Komprimierung der Rechte verlief die Wahl der entsprechenden Darstellungsarten zur Anzeige von Analyseergebnissen. Diese sollten einerseits den Ansprüchen eines Administrators genügen und andererseits in Hinblick auf die Weiterentwicklung des *SAT2 Snap-in Prototyps* als Basis für künftige Auswertungen dienen können.

Deshalb wurden drei spezielle Darstellungsarten entworfen, die diesen Anforderungen (hoffentlich) weitgehend entsprechen, wobei jede eine andere Form der Sicherheitsdarstellung unterstützt. Damit unterscheiden sich auch die Anzahl und das Format der angezeigten Objekte, sowie deren Einfärbung (siehe Kapitel 3.7.6).

Im *SAT2 Snap-in* kann der Benutzer selbst entscheiden, welche der drei Anzeige- bzw. Analyseverfahren er verwenden möchte, indem er im Zuge der Konfiguration einer Abfrage die entsprechende Darstellungsart auswählt (siehe dazu Kapitel 3.7.1.1).

Die folgenden Darstellungsarten stehen derzeit einem Anwender des *SAT2 Snap-ins* zur Verfügung:

- ***Listendarstellung (Flat List):***
  Charakteristisch ist hier, dass kein Verzeichnisbaum aufgebaut wird, sondern alle Knoten mit vollem Verzeichnispfad in der Liste aufscheinen. Die konkrete Darstellungsart wurde unter der Annahme entwickelt, dass für einen Administrator im Zuge einer Sicherheitsanalyse insbesondere alle Abweichungen vom Regelfall interessant sind. Ausgehend von der Menge aller Objekte, auf die der

ausgewählte *Security-Principal* Rechte hat, werden nur jene Objekte angezeigt, die Brüche in der Vererbung bzw. in der Standard-Security aufweisen.

- ***Baumdarstellung (vollständig, Post-Order):***

  Diese Darstellungsart ermöglicht die Anzeige des vollständigen Verzeichnisbaumes. Dabei wird der Baum in *Post-Order*-Reihenfolge [Ott/S.284 ff.], also zuerst alle darunter liegenden Teilbäume und dann die Wurzel, aufgebaut. Durch diese *Bottom-Up* Analyse kann jedes Objekt (Knoten) des Baumes in Abhängigkeit der Eigenschaften seiner Söhne dargestellt werden. Damit ist es z.B. möglich, dem Administrator anzuzeigen, über welche Berechtigungen ein *Security-Principal* „irgendwo" unterhalb eines Knotens maximal verfügt (Anmerkung: vgl. dazu Bruder-Bäume [Ott/S.311 ff.]). Oder man könnte einen Knoten als speziellen „Endknoten" darstellen, wenn alle Objekte unterhalb die gleichen Sicherheitseinstellungen aufweisen. Im Gegensatz zur weiter unten erläuterten Form, der „*On Demand*"-Baumdarstellung, wird in diesem Fall der Baum zu Beginn einer Analyse vollständig aufgebaut. Der Nachteil an dieser Darstellungsart ist die relativ lange Wartezeit zu Beginn einer Analyse, der ein Anwender durch den rekursiven Aufbau des Baumes ausgesetzt wird.

- ***Baumdarstellung („On Demand", Pre-Order):***

  Diese Darstellungsart bietet dem Anwender ebenfalls die Möglichkeit, sich den vollständigen Verzeichnisbaum des zu analysierenden sicherheitsrelevanten Bereiches anzeigen zu lassen. Wesentlich dabei ist aber, dass jeder Teilbaum erst bei Bedarf („*On Demand*") aufgebaut wird, d.h. erst wenn ein Knoten des Baumes vom Anwender in der MMC expandiert wird, werden alle direkten Söhne dieses Knotens aus der Datenbank geholt, berechnet und in die MMC-View eingefügt. Dabei werden in der konkreten Implementierung die einzelnen Objekte (Knoten) entsprechend den Berechtigungen des ausgewählten *Security-Principals* dargestellt. Dieses Verfahren bezeichnen wir als *Top-Down* Analyse. Dabei wird der Baum in *Pre-Order*-Reihenfolge [Ott/S.284 ff.], also die Wurzel vor den darunter liegenden Teilbäumen, aufgebaut. Durch den Aufbau „*On Demand*" erreicht man entsprechende Performance, weil dadurch die Wartezeit beim Erstellen eines (Teil-) Baumes deutlich verkürzt wird.

An dieser Stelle soll auf Kapitel 4 dieser Diplomarbeit verwiesen werden. In dieser abschließenden Fallstudie werden alle drei Darstellungsarten, sowie deren Einfluss auf die Auswertung und Anzeige der einzelnen Objekte, anhand verschiedener Beispiele demonstriert.

3.7.6 Einfärbung der Objekte (Knoten)

Die Einfärbung der einzelnen Objekte bzw. Knoten, welche als Ergebnis einer Analyse angezeigt werden, ist neben der Anzeige der Berechtigungen sicher ein gutes Mittel, um dem Administrator einen schnellen Überblick über die Sicherheitseinstellungen zu geben.

Die *Microsoft Management Console (MMC)* bietet die Möglichkeit, zu jedem in einem in einem *Snap-in* dargestellten Objekt ein Symbol (Bitmap) anzuzeigen. Dies erweist sich als überaus vorteilhaft, um dem Administrator durch gezielte Auswahl von Symbolen und entsprechender Farbgebung Auskunft über den Sicherheitsstatus eines Objektes zu geben. Im *SAT2 Snap-in* werden die jeweiligen Symbole in Abhängigkeit der Zugriffsrechte eines *Security-Principals* eingefärbt.

Die Anzahl der unterschiedlichen Symbole wurde im *SAT2 Snap-in* auf ein Minimum von zwei Symbolen beschränkt. Dabei soll das u.a. aus dem *Windows-Explorer* bekannte Ordner-Symbol () den Anwender darauf aufmerksam machen, dass ein Knoten expandierbar ist und unterhalb weitere Objekte angezeigt werden. Ein Endknoten im Verzeichnisbaum wird durch ein einfaches Datei-Symbol () gekennzeichnet.

Die Anzahl der Farben, die bei der Gestaltung eines *Snap-in* Symbols zur Verfügung steht, wird zwar durch die Anzahl von lediglich 16 Farben, eingeschränkt, ist aber dennoch ausreichend. Im *SAT2 Snap-in Prototyp* wird das Einfärben der Symbole nach dem Ampelprinzip vollzogen. Dazu werden die Farben Rot, Gelb und Grün verwendet, um mehr oder weniger kritische Stellen im System zu kennzeichnen. Die Farbe Grau soll dem Anwender signalisieren, dass ein *Security-Principal* über keine Zugriffsrechte auf ein Objekt verfügt (Anmerkung: Diese Objekte können in einer weiteren Komprimierungsstufe eventuell auch einfach weggelassen werden).

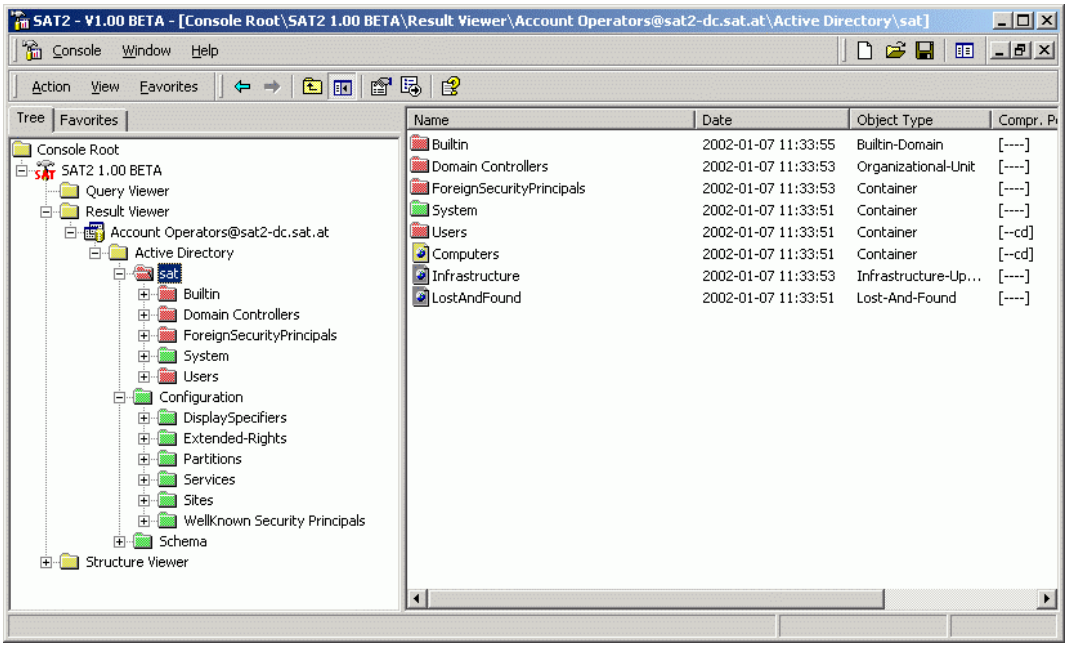


Abbildung 3.24: Symbole und Farben

Abbildung 3.24 soll die verschiedenen Symbole und unterschiedlichen Farben am Beispiel einer *Active Directory* Analyse demonstrieren. Zu beachten ist dabei, dass in diesem Fall die vollständige Baumdarstellung zur Anzeige der Analyseergebnisse gewählt wurde. Den Farben kommt damit eine spezielle Bedeutung zu, die sich nicht nur auf die Zugriffsrechte des untersuchten *Security-Principals* auf das jeweilige Objekt beschränkt.

Wie sich die Darstellungsart auf die Einfärbung der Symbole auswirkt, wird im Folgenden beschrieben.

- ***Listendarstellung und „On Demand“ - Baumdarstellung:***
  Die angezeigten Objekte erhalten ihre Farbe ausschließlich in Abhängigkeit der Zugriffsrechte des *Security-Principals* auf das jeweilige Objekt. Die Bedeutung der Farbe bezieht sich dabei direkt auf das Objekt.
  - Grau: keine Rechte
  - Grün: Leserechte
  - Gelb: Änderungsrechte
  - Rot: Vollzugriff (Full Control)

- ***Vollständige Baumdarstellung:***

  Durch den rekursiven *Post-Order*-Aufbau des Verzeichnisbaumes können bei der Farbgebung alle Zugriffsberechtigungen unterhalb eines Knotens (Objektes) berücksichtigt werden. Aus diesem Grund wird die Farbe in Zusammenhang mit einer *Bottom-Up* Analyse immer von den kumulierten Zugriffsrechten des *Security-Principals* „unterhalb" des einzufärbenden Knotens bestimmt.

  - Grau: keine Rechte
  - Grün: Leserechte oder weniger (!)
  - Gelb: Änderungsrechte oder weniger (!)
  - Rot: Vollzugriff (Full Control)

HINWEIS: Das hier beschriebene Verfahren zum Einfärben der Symbole sowie die Rückschlüsse auf die Zugriffsrechte unter Berücksichtigung der verschiedenen Farben stellen lediglich Beispiele und keine Konventionen, wie Auswertungen zu erfolgen haben, dar. Diese Beispiele sollen dazu dienen, um mit dem *SAT2 Snap-in Prototyp* zu demonstrieren, wie eine Auswertung erfolgen kann. Der Prototyp wurde so konzipiert, dass in künftigen Versionen möglichst einfach neue Symbole, andere Farben oder ein anderes Verfahren zur Färbung der Symbole hinzugefügt werden können. Die Farbe Grau bezieht sich im Falle der vollständigen Baumdarstellung ausschließlich auf Endknoten im Verzeichnisbaum und in der „*OnDemand*"-Baumdarstellung auf alle Objekte, auf welche der *Security-Principal* keine Rechte hat. Im Zuge der Entwicklung des *SAT2 Snap-ins* wurden bereits Überlegungen angestellt, ob es nicht sinnvoller wäre, in einer zusätzlichen Komprimierungsstufe alle grauen Endknoten bzw. alle grauen Knoten, die keine Söhne haben, in der Anzeige vollständig wegzulassen.

3.7.7 Informationen zur Weiterentwicklung

Auf Grund des teilweise unerwartet hohen Aufwandes, den die Implementierung des *SAT2 MMC Snap-ins* mit sich brachte, wurde vom *SAT2* Team – entgegen der ursprünglichen Zielsetzung – beschlossen, mit dem *Snap-in* eine Art Basis-Framework als ersten Prototyp zur Verfügung zu stellen. Diese Framework sollte einerseits genügend Funktionalität enthalten, um erste Auswertungen der Zugriffsrechte im Bereich des *Active Directory* durchführen zu können und andererseits so konzipiert sein, dass vorhandene Analyseverfahren möglichst einfach erweitert oder durch andere ersetzt werden können.

Die Entscheidung, den *SAT2 Snap-in Prototyp* für Analysen im *Active Directory* zu entwickeln, wurde aus mehreren Gründen getroffen. Erstens standen durch die frühzeitige Fertigstellung des *ADS-Scanners* die Daten für entsprechende *Active Directory* Analysen als erste zur Verfügung. Zweitens konnte kein Mitglied des *SAT2* Teams Erfahrungen in Bezug auf Sicherheitsanalysen in diesem Bereich vorweisen. Deshalb sollten die aus den ersten Auswertungen gewonnenen Erkenntnisse als Basis für künftige Implementierungen wegweisend sein. Und nicht zuletzt können alle hier entwickelten Verfahren – in etwas abgeänderter (und eventuell vereinfachter) Form – auch für Analysen des *NTFS 5* oder der *Registry* eingesetzt werden, was umgekehrt auf Grund der höheren Komplexität des *Active Directory* nicht möglich wäre.

Dieses Kapitel soll vorwiegend als Einstiegshilfe für Entwickler dienen, die künftig an diesem Projekt weiterarbeiten, und dem interessierten Leser Auskunft über den aktuellen Stand der Implementierung des *SAT2 Snap-ins* geben.

3.7.7.1 Funktionalität des Frameworks

Der Stand der Implementierung und die Funktionalität, welche das Framework bis dato zur Verfügung stellt, wird im folgenden Abschnitt aufgeschlüsselt.

- ✓ Vollständige Implementierung der Standard und MMC COM Interfaces, Registrierungsfunktionen und des *ISnapinAbout Interfaces* als Grundgerüst des *SAT2 Snap-ins*.

- ✓ Basisklasse zum Ableiten von *Scope Items* und *Result Items*, also jener Objekte, die in den *MMC Namespace* eingefügt werden.

- ✓ Basisklassen für Dialoge und Eigenschaftenfenster.

- ✓ Implementierung aller in der MMC zur Verfügung stehenden GUI Elemente, wie z.B. Symbolleisten, Kontextmenüs, Dialoge und Eigenschaftenfenster.

- ✓ Vollständige Implementierung des *Query Viewers*, dessen Funktionalität das Erstellen, Konfigurieren und Ausführen von Abfragen ermöglicht.

- ✓ Vollständige Implementierung des *Structure Viewers*, der die Struktur des analysierten Netzwerkes repräsentiert und zusätzlich das Erstellen neuer Abfragen ermöglicht.

- ✓ Vollständige Implementierung der Funktionalität zur Analyse der Gruppenmitgliedschaften auf Basis von Benutzern oder Gruppen, jeweils unter Berücksichtigung der direkten und indirekten Mitgliedschaften.

- ✓ Beispielhafte Implementierung der Funktionalität zur Analyse der Zugriffsrechte im *Active Directory* auf Benutzer- oder Gruppenebene, wobei für jeden *Security-Principal* sowohl *Trustee*- als auch effektive Rechte analysiert werden können.
    - Funktionen zur komprimierten Darstellung der Berechtigungen nach dem in Kapitel 3.7.4 beschriebenen Verfahren.
    - Funktionen zur Darstellung der Analyseergebnisse, in Form der drei in Kapitel 3.7.5 beschriebenen Darstellungsarten.
    - Funktionen zur entsprechenden Einfärbung der Symbole in Abhängigkeit der Zugriffsrechte und Darstellungsarten, basierend auf dem in Kapitel 3.7.6 beschriebenen Konzept.
    - Funktionen zur Aufschlüsselung der Zugriffsrechte einzelner Objekte, durch entsprechende Anzeige der ACEs in Eigenschaftenfenstern.

3.7.7.2 Architektur des SAT2 Snap-in

Zur Implementierung *des SAT2 Snap-ins* sollen an dieser Stelle nur wenige detaillierte Hinweise gegeben werden. Alleine der Umfang des Programmcodes ist zu groß, als dass auf einigen wenigen Seiten eine Anleitung zur Weiterentwicklung niedergeschrieben werden könnte. Deshalb sollen zunächst einige Aspekte erläutert werden, die zum Verständnis der Implementierung des *SAT2 Snap-ins* beitragen sollen.

Als Entwicklungsumgebung wurde das *Microsoft Visual Studio 6.0* verwendet und der Programmcode in *C++* geschrieben. Der Grund für diese Entscheidung war, dass einerseits im Microsoft MMC Diskussionsforum [Msng] immer wieder darauf hin gewiesen wird, dass eine Implementierung in *C++* der einzig richtige Weg sei, leistungsfähige *Snap-ins* zu entwickeln. Andererseits wurden sämtliche Programmbeispiele aus [Mmc1] in *C++* entwickelt und konnten dadurch einfacher als Vorlage genutzt werden. Warum auf keine andere Technologie zurückgegriffen wurde, wird auch in Kapitel 3.7.7.4 erläutert.

Die Grundfunktionalität des *SAT2 Snap-in* wurde größtenteils in Anlehnung an die Programmbeispiele aus [Mmc1] und [Mmc2] implementiert. Sollte in diesem Zusammenhang Änderungsbedarf bestehen, wird empfohlen, die beiden o.a. Quellen als Nachschlagewerk zu verwenden. Deshalb wird in diesem Abschnitt auch nicht näher darauf eingegangen.

Erwähnenswert hingegen ist die interne Strukturierung des *SAT2 Snap-ins* und die Vorgehensweise zum Einfügen neuer Objekte (Knoten) in den *MMC Namespace*. Generell wird in der Implementierung immer zwischen *Scope Items* und *Result Items* unterschieden, obwohl alle Objekte bzw. Klassen für Objekte von einer gemeinsamen Basisklasse (*CBaseNode*) abgeleitet werden. Durch eine Implementierung der entsprechenden Methoden einer Klasse wird ihr „Typ“ (*Scope*- oder *Result Item*) festgelegt. Zur Darstellung der Analyseergebnisse wurde eine spezielle Klasse (*CSatObject*) entwickelt, die einerseits von der Basisklasse *CBaseNode* abgeleitet, und andererseits als Basisklasse für weitere Objektklassen verwendet wird. Für *Active Directory* Analysen beispielsweise wird davon jeweils eine Klasse für *Scope Items* und eine für *Result Items* abgeleitet, die wiederum zum Erzeugen der jeweiligen anzuzeigenden Objekte eingesetzt werden.

Abbildung 3.25 zeigt die Objekthierarchie im Sinne der Objektorientierten Programmierung für die Objekte, die im *SAT2 Snap-in* zur Darstellung der Analyseergebnisse in den *MMC Namespace* eingefügt werden. Der Knoten mit der Bezeichnung „*Scope-/Result-Items*“ ist repräsentativ für mehrere verschiedene Objekte gleichen Typs.

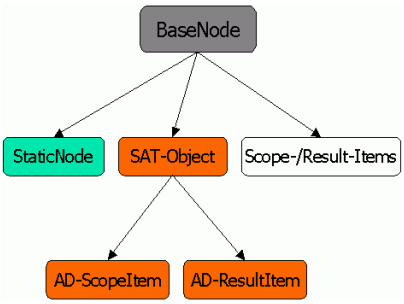


Abbildung 3.25: Objekt-Hierarchie (OOP)

Ein weiteres Kriterium in Bezug auf das Einfügen von Objekten stellt die Tatsache dar, dass immer das Vater-Objekt für das korrekte Erzeugen und Einfügen seiner Sohn-Objekte verantwortlich ist. Dabei ist jedoch zu beachten, dass die zuvor angesprochene Objekthierarchie aus OOP-Sicht nicht mit der Objekthierarchie der MMC gleichzusetzen ist. *Scope Items* können durch entsprechende Implementierung an jeder beliebigen Stelle im *Console Tree* eingefügt werden.

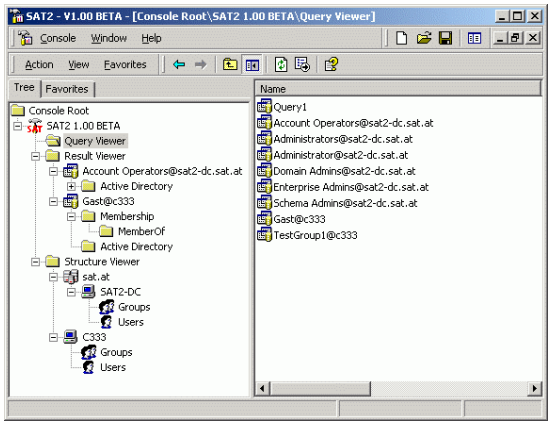


Abbildung 3.26: Objekt-Hierarchie (MMC)

Abbildung 3.26 zeigt den *Console Tree* des *SAT2 Snap-ins*. Die gelb eingefärbten Ordner repräsentieren alle Objekte der selben Objektklasse. Sie würden im Hierarchiebaum in Abbildung 3.25 alle im Knoten „*Scope-/Result-Items*“ anzusiedeln sein.

3.7.7.3 Klassenbeschreibung

Die Beschreibung der implementierten Klassen soll vor allem als Einstiegshilfe für künftige Entwickler dienen. Allerdings kann auch in diesem Fall auf Grund des großen Umfanges nur ein Überblick über die verschiedenen implementierten Klassen, deren Einsatzbereich und Funktionalität, gegeben werden. Weitere Informationen können dementsprechend ausführlich dokumentierten Programmcode entnommen werden.

- ***Klassen: CClassFactory, CDataObject, CComponentData, CComponent***

Die Implementierung dieser COM Klassen ist zwingend erforderlich. Sie bilden das Grundgerüst jedes *MMC Snap-ins*. Die beiden Standard COM Klassen *CClassFactory* und *CDataObject* werden einerseits dazu verwendet, Klassen von COM Objekten zu handhaben und andererseits, um den Datentransfer innerhalb der MMC zu ermöglichen. Die Klasse *CComponentData* stellt die Implementierung des *IComponentData Interfaces* dar, das die Kommunikation der MMC mit dem *Snap-in* im Speziellen mit *Scope Items* ermöglicht. Selbes gilt auch für die Klasse *CComponent*, die ihrerseits für die Verwaltung der *Result Items* sorgt.

- ***Klasse: CSnapinAbout***

Diese Klasse stellt die Implementierung des *ISnapinAbout Interfaces* dar. Sie ermöglicht die Anzeige von Informationen über Hersteller und Versionsnummer, sowie einer Kurzbeschreibung des *Snap-ins* im „*Snap-in hinzufügen/entfernen*“ – Dialog. Zusätzlich kann durch eine Implementierung entsprechender Methoden, das Symbol des *Snap-in* Wurzelknotens (*Static Node*) nach belieben geändert werden.

- ***Klasse: CBaseNode***

Virtuelle Basisklasse, von der alle weiteren Klassen für *MMC Namespace* Objekte abgeleitet werden. Sie enthält die Definitionen aller Methoden, die den Datenaustausch zwischen den einzelnen Objekten und der MMC ermöglichen.

- ***Klasse: CBaseDialog***

Basisklasse für nicht-modale Dialoge, die in *Snap-ins* deshalb verwendet werden müssen, weil modale Dialoge den *UI Thread* blockieren und so den Nachrichtenaustausch

zwischen MMC und anderen *Snap-ins* unterbinden würden, bis der modale Dialog wieder geschlossen wird.

- ***Klasse: CBasePropertyPage***

Die Funktionalität der einzelnen Seiten eines Eigenschaftenfensters in *MMC Snap-ins* wird durch die Implementierung verschiedener Klassen für die jeweiligen Seiten bestimmt. Zu diesem Zweck wurde eine Basisklasse für Eigenschaftenseiten entwickelt, von der alle Klassen für spezielle Eigenschaftenseiten abgeleitet werden können.

- ***Klasse: CSatObject***

Diese Klasse wurde speziell für Objekte entwickelt, die im Zuge einer Sicherheitsanalyse im *Snap-in* angezeigt werden. Sie wird einerseits von der Klasse *CBaseNode* abgeleitet und dient andererseits wiederum als Basisklasse für sowohl *Scope*- als auch *Result Items*, aus denen sich die Darstellung der Analyseergebnisse zusammensetzt. Aus diesem Grund enthält die Klasse *CSatObject* die Funktionalität zur Auswertung der ACLs/ACEs, Komprimierung der Rechte, Einfärbung der Symbole und Initialisierung eines Objektes mit den Daten aus der *SAT2* Datenbank.

- ***Klasse: CStaticNode***

Der Wurzelknoten jedes *Snap-ins* wird als *Static Node* bezeichnet. Alle anderen Objekte im *MMC Namespace* sind Söhne dieses statischen Knotens im Sinne der MMC-Objekt-Hierarchie, werden aber aus Sicht der Implementierung nicht davon abgeleitet. Der *Static Node* wird automatisch eingefügt, wenn das *Snap-in* geladen wird und ist solange präsent, bis das *Snap-in* wieder entfernt wird. Im *SAT2 Snap-in* sorgt der *Static Node* zusätzlich für das Erstellen und Einfügen der drei Hauptkomponenten: *Query Viewer*, *Result Viewer* und *Structure Viewer*.

- ***Klassen: CQueryViewer, CQuery***

Der *Query Viewer* erhält seine Funktionalität durch die Implementierung dieser beiden Klassen. Objekte der Klasse *CQuery* stellen die einzelnen Abfragen dar, die vom Anwender erstellt wurden und enthalten demnach sämtliche Konfigurationsdaten, die zum Durchführen einer Sicherheitsanalyse benötigt werden. Der *Query Viewer*-Container hingegen ist ein Objekt der Klasse *CQueryViewer* und ist für die Verwaltung (Erstellen,

Einfügen und Anzeigen) der *CQuery*-Objekte verantwortlich. Diese Klasse enthält auch die Funktionalität zum Aufrufen des „New Query“ – Dialogs.

- ***Klasse: CQueryNewDlg***

Implementierung des Dialogs zum Anlegen neuer Abfragen. Der „New Query“ – Dialog wird aus dem *Query Viewer* aufgerufen und ermöglicht dem Anwender die Konfiguration jener Einstellungen, die für eine Sicherheitsanalyse benötigt werden. Die vom Anwender spezifizierten Daten werden an den *Query Viewer* weitergeleitet und dienen als Basis zum Initialisieren der *CQuery*-Objekte. Diese Klasse wird von der *CBaseDialog* Klasse abgeleitet.

- ***Klassen: CQueryPropPageGeneral, CQueryPropPageOptions, CQueryPropPageAdcanced***

Um die Benutzeroberfläche möglichst flexibel zu gestalten besteht die Möglichkeit, die Konfigurationsdaten der verschiedenen Abfragen unter Verwendung eines Eigenschaftenfensters einzusehen und zu verändern. Jede Eigenschaftenseite wird als eigene Klasse implementiert und erbt ihre Grundfunktionalität von der Basisklasse *CBasePropPage*. Aufgerufen wird ein Eigenschaftenfenster vom jeweiligen *CQuery*-Objekt.

- ***Klasse: CStructureViewer***

Der *Structure Viewer* soll die Struktur des analysierten Netzwerks repräsentieren. Zu diesem Zweck wird zuerst ein Container-Objekt der Klasse *CStructureViewer* als direkter Sohn des *Static Nodes* eingefügt. Von hier aus werden alle Datenbankabfragen, die Informationen zur Struktur des Netzwerks liefern sollen, abgesetzt. Auf Basis dieser Daten werden dann Objekte der Klassen *CDomainFolder* und/oder *CComputerFolder* erzeugt und als Söhne des *Structure Viewers* eingefügt.

- ***Klasse: CDomainFolder***

Objekte der Klasse *CDomainFolder* bezeichnen die verschiedenen Domains eines gescannten Netzwerks. Sie dienen dazu, alle Rechner in einem Netzwerk in Abhängigkeit ihrer Domain-Zugehörigkeit zu gruppieren. Ein Domain-Container enthält demnach alle Computer, die einer bestimmten Domäne angehören.

- ***Klasse: CComputerFolder***

Ein Computer-Container kann entweder als Sohn eines Domain-Containers oder als direkter Sohn des *Structure Viewers* eingefügt werden, wenn er keiner Domäne zugeordnet werden kann und bezeichnet jeweils einen Computer, der im Zuge des *Security-Scans* analysiert wurde. In jedem Fall erzeugt der jeweilige Computer-Container Objekte der Klassen *CGroupFolder* und *CUserFolder*, die dazu verwendet werden, um Gruppen und Benutzer eines Computers zu gruppieren.

- ***Klassen: CGroupFolder, CUserFolder***

Nachdem ein Gruppen- bzw. Benutzer-Container erstellt wurde, erfolgen von hier aus jene Datenbankabfragen, die Informationen über die verschiedenen Gruppen und Benutzer liefern sollen, welche auf einem bestimmten Computer ausgelesen wurden. Auf Basis dieser Daten werden Objekte der Klassen *CGroup* und/oder *CUser* erzeugt, mit den Daten aus der *SAT2* Datenbank initialisiert und in den jeweiligen Container eingefügt.

- ***Klassen: CGroup, CUser***

Objekte dieser Klassen repräsentieren bestimmte Gruppen bzw. Benutzer. Sie enthalten alle Informationen die in der *SAT2* Datenbank gespeichert wurden und sollen dem Anwender Aufschluss über die Eigenschaften des jeweiligen *Security-Principals* geben. Aus Gründen der Benutzerfreundlichkeit kann über das mit dem Objekt assoziierte Kontext-Menü der „New Query“ – Dialog, zum Erstellen einer Abfrage für das selektierte *Security-Principal*, aufgerufen werden.

- ***Klassen: CResultViewer, CQueryResultFolder***

Der *Result Viewer* stellt lediglich einen Container dar, in dem alle durchgeführten Analysen bzw. deren Ergebnisse „gesammelt“ werden. Für jede durchgeführte Abfrage wird vom *Result Viewer* aus ein Objekt der Klasse *CQueryResultViewer* erzeugt und eingefügt. Diese Objekte stellen einerseits die verschiedenen Abfragen dar und müssen andererseits dafür Sorge tragen, dass entsprechend den Konfigurationsdaten jene Container eingefügt werden, welche die Ergebnisse von Mitgliedschafts- bzw. *Active Directory* Analysen enthalten. Ansonsten verfügen beide Klassen über keine besondere Funktionalität.

- ***Klassen: CMembershipFolder, CMemberOfFolder, CMembersFolder, CMembership***

Diese vier Klassen stellen sämtliche Funktionen zur Verfügung, die für eine Analyse der Gruppenmitgliedschaften erforderlich sind. Dabei sorgt der Membership-Container (*CMembershipFolder*) für das Erstellen der Container-Objekte der Klassen *CMemberOfFolder* und/oder *CMembersFolder*. Diese beiden Container führen die eigentliche Analyse der Mitgliedschaften in Form gezielter Datenbankabfragen durch und erzeugen Objekte der Klasse *CMembership*, welche die eigentlichen Analyseergebnisse repräsentieren.

- ***Klasse: CActiveDirectoryFolder***

Der *Active Directory*-Container enthält die jeweiligen Ergebnisse einer *Active Directory* Analyse, bildet also die Basis für alle Analyseergebnisse. In Abhängigkeit der Konfiguration einer Abfrage werden die entsprechenden Datenbankabfragen von dieser Stelle aus initiiert. Für jeden Datensatz, den diese Abfragen liefern, wird jeweils ein entsprechendes Objekt der Klassen *CADScopeItem* bzw. *CADResultItem* erzeugt und in den *MMC Namespace* eingefügt. Der Typ eines Objektes legt in diesem Fall fest, ob ein *Scope Item* – also ein Container – oder ein *Result Item* – ein Endobjekt – erzeugt werden muss.

- ***Klassen: CADScopeItem CADResultItem***

Beide Klassen erben ihre Funktionalität von der speziell für *Active Directory* Analysen entwickelten Basisklasse *CSatObject*. Diese Funktionalität umfasst neben dem Initialisieren des Objekts auch Methoden zur Auswertung der Sicherheitseinstellungen anhand der Informationen aus der *SAT2* Datenbank, deren (komprimierte) Darstellung und entsprechende Einfärbung der Symbole.

- ***Klasse: CADObjectPropPageGeneral***

Um dem Anwender die Möglichkeit zu bieten, sich detaillierte Informationen über die Sicherheitseinstellungen eines Objektes anzeigen zu lassen, die eventuell durch die komprimierte Darstellung der Rechte im *Result Pane* verloren gehen, wurde ein spezielles Eigenschaftenfenster entwickelt. In diesem Eigenschaftenfenster werden alle ACEs aufgeschlüsselt, welche den ausgewählten *Security-Principal* betreffen. Die einzelnen

Zugriffsrechte, die aus den jeweiligen ACEs resultieren, werden dabei exakt dargestellt und sollen so Aufschluss über den Sicherheitsstatus eines bestimmten Objektes geben.

3.7.7.4 Schwierigkeiten und Probleme

Dieser Punkt soll vorwiegend dazu dienen, künftige *Snap-in* Entwickler auf mögliche Probleme hinzuweisen, mit denen jederzeit zu rechnen ist. Zu diesem Zweck sollen kurz einige der Schwierigkeiten aufgezeigt werden, mit denen der Autor dieser Arbeit im Zuge der Implementierung des *SAT2 Snap-ins* konfrontiert war.

Zu Beginn soll natürlich der extrem hohe Aufwand, der mit der Entwicklung eines *Snap-ins* verbunden ist, erwähnt werden. Er bezieht sich sowohl auf die Einarbeitung als auch auf die Implementierung. Entsprechende Kenntnisse über das *MMC Framework* und dessen Funktionalität sind jedoch Grundvoraussetzungen, um *Snap-ins* entwickeln zu können. Der Aufwand für die Implementierung ist sicherlich abhängig von der Funktionalität, über die ein *Snap-in* letztendlich verfügen soll, ist aber auch für *Snap-ins* mit wenig Funktionalität nicht unbeachtlich.

Weiters existieren zum Thema *Snap-in* Entwicklung so gut wie keine Referenzen. Neben den Informationen aus dem *Microsoft Platform SDK* [Mmc2] stand lediglich ein Buch – *Microsoft MMC Design und Development Kit* [Mmc1] – als Nachschlagewerk zur Verfügung. Beide Quellen vereinfachen die Entwicklung nur bedingt. Die brauchbarsten Informationen erhält man aus der *Microsoft MMC Newsgroup* [Msng], die im übrigen das einzige (dem Autor bekannte) Diskussionsforum für *Snap-in* Entwickler darstellt.

Erschwerend hinzu kam, dass die Programmbeispiele aus dem *Platform SDK* nicht nur äußerst trivial, sondern teilweise auch fehlerhaft sind. Aus diesem Grund waren diese Beispiele nur beschränkt hilfreich und die Verwendung des Programmcodes als Basis zur Implementierung des *Snap-in* Grundgerüsts nur teilweise möglich.

Mit gravierenden Schwierigkeiten sieht sich ein Entwickler auch dann konfrontiert, wenn er versucht ein *MMC Snap-in* mit MFC Funktionalität auszustatten. Die entsprechenden Programm-Bibliotheken einzubinden ist schlichtweg unmöglich. Um diese

Funktionalität nicht „von Hand“ hinzufügen zu müssen, empfiehlt Microsoft deshalb, den *ATL/COM Wizard* zu verwenden. Aber ...

Der *ATL/COM Application Wizard* wurde seit geraumer Zeit nicht mehr weiterentwickelt und wird künftig aus der Microsoft Produktpalette gestrichen [Msng1]. Deshalb wird von seiner Verwendung abgeraten. Grundsätzlich kann dieser *Wizard* in der *Snap-in* Entwicklung zwar noch eingesetzt werden, der Entwickler ist dabei jedoch auf sich alleine gestellt und darf nicht mit einer Unterstützung seitens *Microsoft Support* rechnen.

Ähnlich verhält es sich auch mit dem *VB Snap-in Designer*. Abgesehen von der Tatsache, dass der *VB Designer* für komplexere Aufgabenstellungen nicht geeignet scheint, wird er voraussichtlich künftig nicht mehr weiterentwickelt werden [Msng2].

Dies sind nur einige der Schwierigkeiten oder Probleme, mit denen sich *Snap-in* Entwickler auseinandersetzen müssen. Diese Liste könnte nahezu endlos fortgesetzt werden. Dabei soll aber keineswegs der Eindruck entstehen, dass die MMC ein minderwertiges Produkt sei oder Microsoft nur mangelnde Unterstützung für ihre Produkte anbietet. Dieser Abschnitt soll lediglich darauf hinweisen, dass die Entwicklung von *MMC Snap-ins* nicht annähernd so einfach ist, wie sie in [Mmc1] dargestellt wird.

3.8 Ausblick

Abschließend soll dem Leser ein Ausblick auf künftige Vorhaben des *SAT2* Teams und mögliche Erweiterungen gewährt werden.

Das erste Vorhaben, welches geplant war und mittlerweile bereits realisiert wird, ist die Implementierung des *Controllers* als *MMC Snap-in*. Durch den Einsatz einer graphischen Benutzeroberfläche zur Konfiguration der verschiedenen Datensammler in Form eines *MMC Snap-ins* soll die Durchführung eines *Security-Scans* beträchtlich vereinfacht werden.

Darüber hinaus ist sich *SAT2* Team gewisser Schwächen des *Security Analysis Tools 2 (SAT2)* durchaus bewusst, einige davon mussten auf Grund bestimmter Gegebenheit, auf die hier nicht näher eingegangen werden soll, in Kauf genommen werden, andere resultieren aus manchen Kompromisslösungen, die vor allem aus Zeitgründen eingegangen werden mussten. Diese Mängel sollen jedoch in künftigen Versionen Schritt für Schritt behoben werden.

Die wahrscheinlich größte Schwachstelle des *SAT2* Systems liegt in der Performance. Auf Grund der hohen Datenmenge, die sowohl gespeichert, als auch zu Analysezwecken ausgelesen werden muss, entstehen für den Anwender u.U. zu lange Wartezeiten. Um dem entgegenzuwirken, soll das Datenbankkonzept in künftigen Versionen neu überarbeitet und dadurch verbessert werden.

Ein weiteres Vorhaben zur Erweiterung des *SAT2* Systems ist im Bereich der *Security-Scanner* geplant. Dabei sollen künftig sogenannte SELF und OWNER Analysen (optional) durchgeführt werden können, indem die SIDs dieser *Security-Principals*, in diesem Fall die Besitzer eines Objektes, im Zuge der Sicherheitsanalyse berücksichtigt werden.

Ebenfalls geplant ist die Weiterentwicklung des *SAT2 Snap-in Prototyps*. Hier sollen dem Anwender zunächst Analysen der Zugriffsrechte im *NTFS* und im Bereich der *Windows-Registry* ermöglicht werden. Außerdem ist geplant, auch im *SAT2 Snap-in* Komprimierungsverfahren einzusetzen, die beispielsweise das Zusammenfassen von verschiedenen Objekten mit gleichen Sicherheitseinstellungen und deren Anzeige durch ein repräsentatives Objekt realisieren sollen. Auch sollen neue Konzepte zur Auswertung der Sicherheitseinstellungen entworfen werden. Dadurch sollen u.a. Abfragen wie z.B. „Wo haben alle Benutzer mit gleichem Namen Rechte?“ oder „Wo haben alle Mitglieder einer bestimmten Gruppe Rechte?“ durchgeführt werden können.

4 Fallstudie – Das SAT2 MMC Snap-in

Die abschließende Fallstudie soll vorwiegend dazu dienen, dem Leser die Funktionalität des *SAT2 Snap-ins* zu demonstrieren. Im vorangegangenen Kapitel wurden jene Konzepte beschrieben, welche die Basis für Analysen der Zugriffsrechte bilden und die entsprechende Darstellung der Ergebnisse ermöglichen. Anhand ausgewählter Beispiele werden verschiedenen Analyseverfahren näher erläutert. Dies soll vor allem zum besseren Verständnis der Grundkonzepte beitragen.

4.1 Ausgangsbasis

Der Prototyp des *SAT2 Snap-ins* wurde speziell für Auswertungen im Bereich des *Active Directory* konzipiert. Zu Demonstrationszwecken wurde die trivialste Form einer Arbeitsumgebung geschaffen, die möglich war, um Analysen im *Active Directory* durchführen zu können.

Als Betriebssystem wurde der *Microsoft Windows 2000 Advanced Server* installiert. Dieser Server wurde anschließen im Zug der Installation des *Active Directory* (Domain: sat.at), zu einem Domain-Controller (NetBIOS Name: SAT2-DC) designiert. Zusätzlich musste natürlich ein Datenbankserver installiert werden, um die aus dem *Security-Scan* resultierenden Daten entsprechend archivieren zu können. Zu diesem Zweck wurde der *Microsoft SQL Server 2000* verwendet.

Der Grund, warum diese einfache Konstellation zur Vorführung des *SAT2 Snap-ins* hinreichend ist, liegt in der Natur des *Active Directory*. Objekte im *Active Directory* werden auf alle Domain-Controller innerhalb einer Domain repliziert. Das *Active Directory Schema* wird auf alle Domain-Controller innerhalb des gesamten *Active Directory* repliziert. Sinngemäß werden *Active Directory Objekte* demnach auf jeweils einem DC pro Domain und das *Active Directory Schema* auf einem DC im gesamten *Active Directory* durchgeführt.

4.2 Security-Scan

Als Basis für alle folgenden Auswertungen wurde in der zuvor beschriebenen Arbeitsumgebung mit der Datensammler-Komponente des *SAT2* Systems ein *Security-Scan* durchgeführt.

Der Ablauf dieses *Security-Scans* im Testsystem gestaltete sich folgendermaßen. Unter Einhaltung der „vorgeschriebenen“ Reihenfolge wurden die einzelnen Komponenten des Datensammlers, jeweils mit vollem Funktionsumfang, gestartet.

1) ***Controller/Master***

   AUFGABE: Erstellen der *SAT2* Datenbank und zugehöriger Tabellen.

   ANMERKUNG: Der (derzeitige) zu Testzwecken verwendete *Controller* ist eine einfach Kommandozeilen-Applikation, welche die *SAT2* Datenbank, entsprechend dem derzeit verwendeten Datenbankkonzept, erstellt.

2) ***CGU-Scanner***

   AUFGABE: Computer-, Gruppen-, und Benutzeranalyse und ermitteln der Gruppenmitgliedschaften.

   ANMERKUNG: Auch der *CGU-Scanner* ist derzeit noch eine eigenständige Kommandozeilen-Applikation, die nach der Fertigstellung des *NTFS/REG-Scanners* gemeinsam mit dem *ADS-Scanner* in einen *Service* „verpackt“ werden wird.

3) ***ADS-Scanner***

   AUFGABE: Analyse des DomainNC-Containers, Configuration-Containers und Schema-Containers. Komprimierungsstufe wurde keine gewählt.

   ANMERKUNG: Nach [Hel] ist der *ADS-Scanner* derzeit sowohl als *Service*, als auch als Kommandozeilen-Applikation lauffähig. Dieser Service kann sich – nach einer Konfiguration der entsprechenden *Switches* durch den Anwender – quasi „selbst“ installieren.

Nach Beendigung des *Security-Scans* umfasst die *SAT2* Datenbank u.a. die in Tabelle 4.1 aufgeschlüsselten Daten.

Daten	Anzahl
Active Directory Objekte	2439
ACLs	64
ACEs	632
Gruppen	34
Benutzer	9
Mitgliedschaften	41

Tabelle 4.1: Datensätze in der SAT2 Datenbank

Auf Basis dieser Daten wurden eine Reihe von Auswertungen durchgeführt, welche auf den folgenden Seiten erläutert werden sollen.

4.3 Analysen mit dem SAT2 Snap-in

Der Dokumentation des *SAT2 Snap-ins* in Kapitel 3.7.1 kann entnommen werden, welche Schritte erforderlich sind, um gewünschte Abfragen zu konfigurieren und auszuführen. Deshalb soll in diesem Abschnitt nicht mehr darauf eingegangen werden. Das Hauptaugenmerk ist hier eher auf die Ergebnisse einer Analyse gerichtet.

Im Zuge der Implementierung und Tests des *SAT2 Snap-ins* wurden viele Auswertungen der Zugriffsrechte im Bereich *Active Directory* durchgeführt. Speziell für diese Fallstudie wurden jedoch vier verschiedene Szenarien entworfen. Diese sollen vor allem die Auswirkungen der unterschiedlichen Analyseverfahren auf die Darstellung der Ergebnisse zeigen und somit wesentlich zum Verständnis des *SAT2 Snap-ins* beitragen.

1) Analyse der Gruppenmitgliedschaften
2) Analyse von Brüchen in Standard-Security und Vererbung (*Flat List*)
3) *Top-Down* Analyse (*Pre-Order* Baumdarstellung)
4) *Bottom-Up* Analyse (*Post-Order* Baumdarstellung)

4.3.1 Szenario 1 – Analyse der Mitgliedschaften

Am Beispiel des Benutzers „Administrator“ soll die Analyse der Gruppenmitgliedschaften demonstriert werden. Zu diesem Zweck wurde eine entsprechende Abfrage unter Verwendung des „New Query“ – Dialogs erstellt und gestartet.

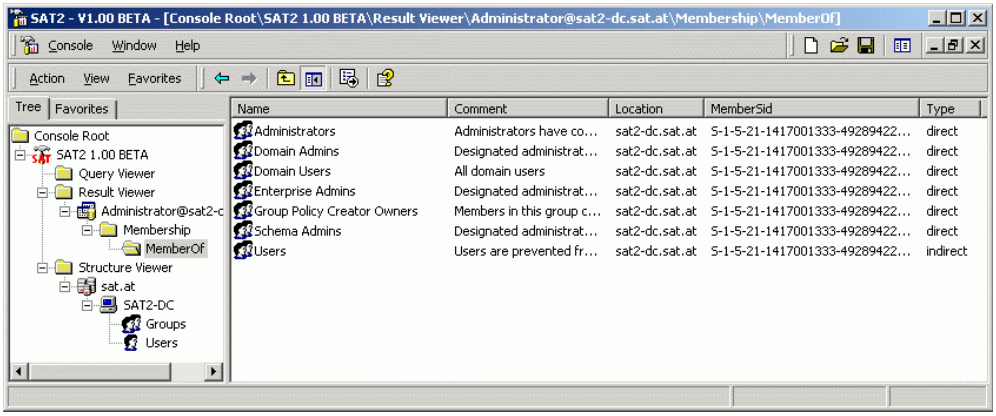


Abbildung 4.1: Membership Analyse

Eine Analyse der Mitgliedschaften gibt Auskunft über die Gruppenzugehörigkeit eines *Security-Principals*. Dabei werden sowohl direkte als auch indirekte Mitgliedschaften sowie Informationen über die entsprechenden Gruppen oder Benutzer (siehe Abbildung 4.1) angezeigt. Nützlich erweist sich diese Auswertung vor allem in Kombination mit einer *Active Directory* Analyse. Sollen beispielsweise die effektiven Rechte eines Benutzers ermittelt werden, kann man auf diesem Weg am besten feststellen, welche Gruppen in diese Auswertung mit einbezogen werden.

4.3.2 Szenario 2 – Brüche in Standard-Security und Vererbung

Szenario 2 soll am Beispiel der Gruppe „Account Operators“ die einfachste Variante der drei implementierten Darstellungsarten bzw. Analyseverfahren demonstrieren.

Das Ziel einer derartigen Anzeige der Berechtigungen ist, dem Anwender alle Abweichungen vom Regelfall zu präsentieren. Diese Abweichungen beziehen sich dabei ausschließlich auf die Standard-Security und Vererbung. Weisen die Sicherheitseinstellungen eines Objektes beispielsweise Abweichungen gegenüber der Standard-Security auf oder wird die Vererbung durch ein Objekt unterbrochen, wird dieses angezeigt und der

vorliegende Bruch in der Spalte „Breaks Inh./StdSec.“ gekennzeichnet. Zusätzlich werden auch in dieser Darstellungsart die Berechtigungen, des *Security-Principals* – in diesem Fall der Gruppe „Account Operators“ – auf das jeweilige Objekt angezeigt. Die Berechtigungen werden dabei in dem, in Kapitel 3.7.4 beschriebenen, komprimierten Format angezeigt.

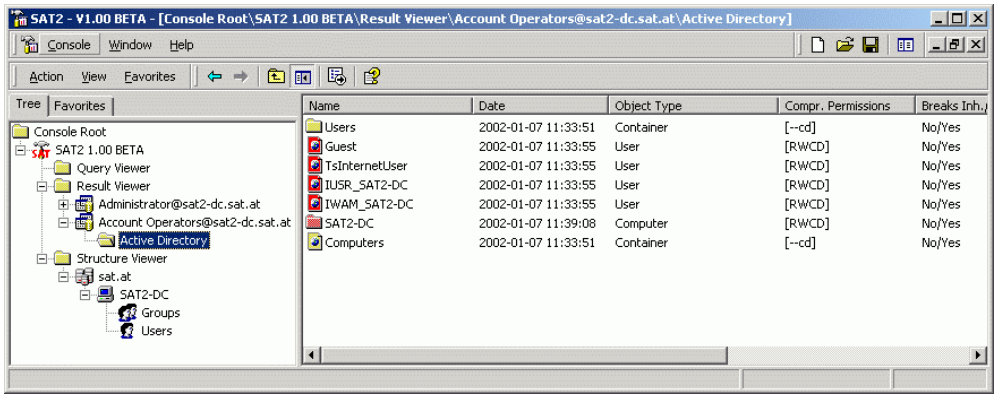


Abbildung 4.2: Anzeigen von Ausnahmen

Die Vorteile dieses Analyseverfahrens liegen auf der Hand. Trotz der hohen Datenmenge können Auswertungen mit diesem Verfahren äußerst schnell durchgeführt werden. Durch gezielte SQL-Abfragen, wird die Datenmenge auf ein Minimum reduziert und besitzt dennoch genügend Aussagekraft um mögliche Schwachstellen in einem System aufzuzeigen. Ein Nachteil – wenn überhaupt – ist, dass mit diesem Verfahren oft auch Objekte angezeigt werden, die in Wahrheit keine „Gefahr“ bedeuten. Der Grund dafür ist, dass im Zuge der Auswertung kein Vergleich zur Standard-Security mehr möglich ist und deshalb nicht geprüft werden kann, in wie weit diese Abweichung eine Bedrohung der Sicherheit eines Systems darstellt.

Betrachtet man in diesem Zusammenhang beispielsweise die einzelnen Berechtigungen des Benutzers „Guest“ – unter zu Hilfenahme eines Standard-Tools – genauer, so lassen sich daraus folgende Erkenntnisse gewinnen. Dem o.a. Benutzer wurde die Berechtigung entzogen, sein Passwort zu ändern. Dies steht allerdings in Widerspruch zur Standard-Security des allgemeinen Benutzerobjektes, von welchem der „Guest“-Benutzer bzw. seine Zugriffsrechte abgeleitet wurde(n). Deshalb weist dieses Objekt einen Bruch in der Standard-Security auf und wird im Zuge der Auswertung angezeigt.

4.3.3 Szenario 3 – Top-Down Analyse

Um den Unterschied zur Darstellung der Ergebnisse im Szenario 2 deutlich zu machen, wurde die im Folgenden beschriebene Auswertung wiederum für die Gruppe der „Account Operators“ durchgeführt.

Im Falle der *Top-Down* Analyse wird der vollständige Verzeichnisbaum des *Active Directory* „*On Demand*“ aufgebaut. Das bedeutet, dass erst, wenn ein Knoten im Verzeichnisbaum der MMC (*Console Tree*) expandiert wird, die direkten Söhne dieses Knotens eingefügt werden. Die Struktur des *Active Directory* wird dabei anhand der Informationen aus der Datenbank nachgebildet. Jene Objekte, auf die der ausgewählte *Security-Principal* Zugriffsrechte besitzt, werden entsprechend gekennzeichnet (siehe Kapitel 3.7.6).

Dieses Verfahren ermöglicht dem Anwender einen vollständigen Einblick in die Struktur des *Active Directory* und verkürzt die Wartezeit zu Beginn der Auswertung (im Gegensatz zur *Bottom-Up* Analyse) beträchtlich. Der Haken an der Sache ist, dass der Anwender u.U. den gesamten Verzeichnisbaum inspizieren muss, um – im *worst case* – letztendlich festzustellen, dass ein *Security-Principal* keinerlei Zugriffsrechte besitzt.

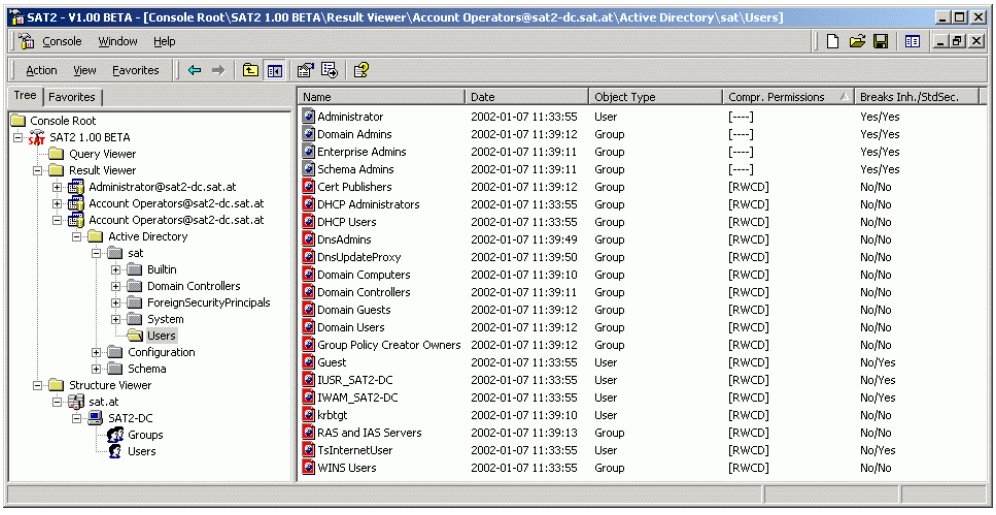


Abbildung 4.3: Top-Down Analyse

Abbildung 4.3 zeigt einen Bildschirmausschnitt der *Top-Down* Analyse für die Gruppe „Account Operators“. Im *Scope Pane* des *Snap-in* sieht man einen Teilbaum des *Active Directory*. Die Farbe der Symbole (sieh auch Kapitel 3.7.6) bezieht sich dabei direkt auf das Objekt, unabhängig von den Rechten „unterhalb“. Offensichtlich hat die Gruppe der „Account Operators“ in diesem Teil des *Active Directory* ausschließlich Rechte auf den Users-Container. Das gelbe Symbol bezeichnet in diesem Fall Änderungsrechte. Vergleicht man dazu Abbildung 4.2, lässt sich aus der Darstellung der Berechtigung (Users: Compr. Permissions = [--cd]) erkennen, dass die Gruppe tatsächlich über Rechte zum Erstellen ([c]) bzw. Löschen ([d]) von bestimmten Objekten dieses Containers verfügt. Man beachte in diesem Zusammenhang die Kleinschreibung der beiden Buchstaben (Klein bedeutet immer „nicht alles“). Diese resultiert daraus, dass ein *Account Operator* im Users-Container lediglich User- und Group-Objekte anlegen bzw. löschen darf (siehe auch Abbildung 4.5).

Der Inhalt des Users-Containers setzt sich aus Benutzer- und Gruppenobjekten zusammen. Anhand der Farben bzw. der Rechtedarstellung kann man erkennen, dass die Gruppe „Account Operators“ auf die meisten Objekte Vollzugriff (Farbe: Rot; Rechte: [RWCD]) besitzt, auf die drei Hauptgruppen (wie z.B. die Gruppe „Domain Admins“) und den Benutzer „Administrator“ aber typischerweise keine Rechte (Farbe: Grau; Rechte: [----]). Dies entspricht genau dem – in Abbildung 4.3 dargestellten – zu erwartenden Ergebnis. (Anmerkung: Es ist sicher nicht erwünscht, dass ein *Account-Operator* sich selbst zur Gruppe der „Domain Admins“ einträgt und so über die Mitgliedschaft in dieser Gruppe auch automatisch Mitglied der Gruppe „Administrators“ wird. Damit hätte faktisch jeder *Account-Operator* die Möglichkeit, sich zum „vollen“ Administrator zu machen. Somit wäre die Gruppe „Account Operators“ eigentlich überflüssig.)

4.3.4 Szenario 4 – Bottom-Up Analyse

Zuletzt soll noch das dritte Analyseverfahren erläutert werden. Auch dafür eignet sich die Gruppe der „Account Operators“. Im Unterschied zu den beiden zuvor beschriebenen Methoden, wird hier der vollständige Verzeichnisbaum des *Active Directory* zu Beginn einer Auswertung rekursiv aufgebaut.

Aus den kumulierten Berechtigungen des ausgewählten *Security-Principals* „unterhalb“ eines Knotens ergibt sich bei der *Bottom-Up* Analyse die entsprechende Farbe des mit einem Objekt assoziierten Symbols (siehe Kapitel 3.7.6). Die komprimierte Rechtedarstellung bezieht sich allerdings wieder auf das Objekt selbst.

Als problematisch stellte sich jedoch die relativ lange Wartezeit zu Beginn der Auswertung heraus, die aus dem *Post-Order*-Aufbau des Verzeichnisbaums, insbesondere bei einer großen Zahl von Objekten, resultiert.

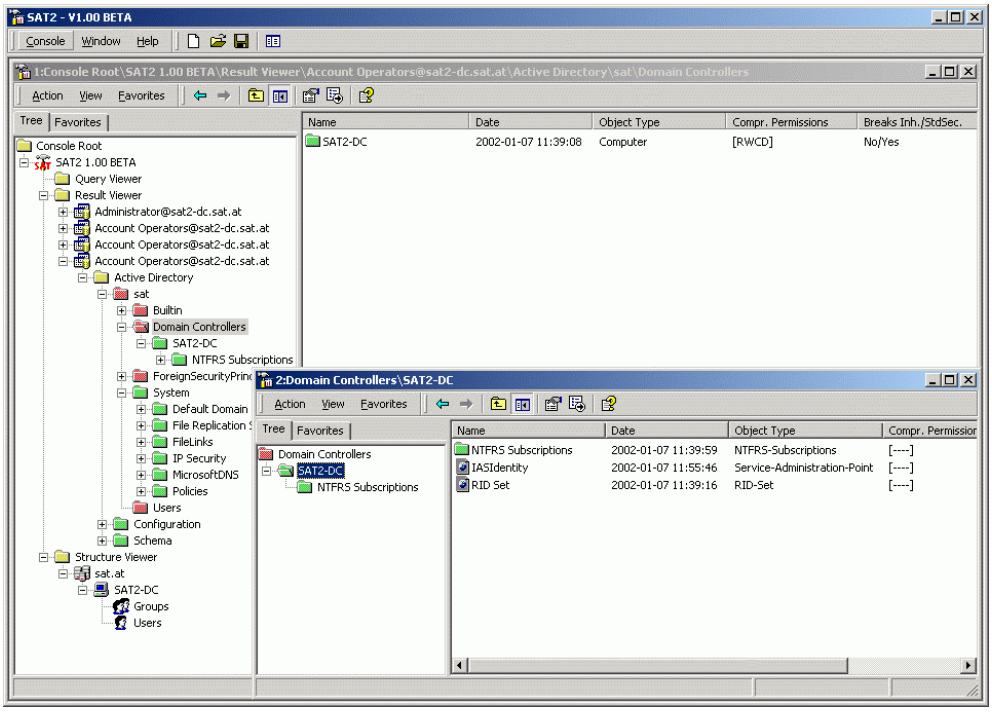


Abbildung 4.4: Bottom-Up Analyse

Abbildung 4.4 zeigt die Auswirkungen des *Bottom-Up* Analyseverfahrens auf die Farbe der Objekte. Obwohl die Gruppe der „Account Operators“ auf den Container mit der Bezeichnung „SAT2-DC“ Vollzugriff besitzt, wurde das Container-Symbol grün eingefärbt. Grün bedeutet in diesem Zusammenhang: Leserechte oder weniger. Das zweite Fenster in Abbildung 4.4 soll dieses Prinzip verdeutlichen. Für alle direkten Söhne des SAT2-DC-Containers gilt: die o.a. Gruppe besitzt keine Rechte für diese Objekte (Farben: Grau bzw. Grün; Rechte: [----]). Die Väter von „SAT-DC“ hingegen sind alle rot markiert, denn auf „SAT-DC“ besitzt der *Security-Principal* alle Rechte.

Um wieder auf den Users-Container aus Szenario 3 (Abbildung 4.2) zurückzukommen, dieser Container wurde im Zuge der *Bottom-Up* Analyse rot eingefärbt, obwohl die Gruppe der „Account Operators“ lediglich Änderungsrechte für dieses Objekt besitzt. Der Grund dafür ist offensichtlich. Wie Abbildung 4.3 entnommen werden kann, verfügt ein *Account Operator* über Vollzugriff auf einige Objekte innerhalb des Users-Containers, der genau aus diesem Grund mit einem roten Symbol versehen wurde.

Wie in Kapitel 3.7.4 erwähnt besteht für den Anwender die Möglichkeit, über ein Eigenschaftenfenster alle durch die komprimierte Darstellung der Rechte verlorengegangen Informationen einzusehen. Zu diesem Zweck werden die den *Security-Principal* betreffenden ACEs in diesem Eigenschaftenfenster aufgelistet.

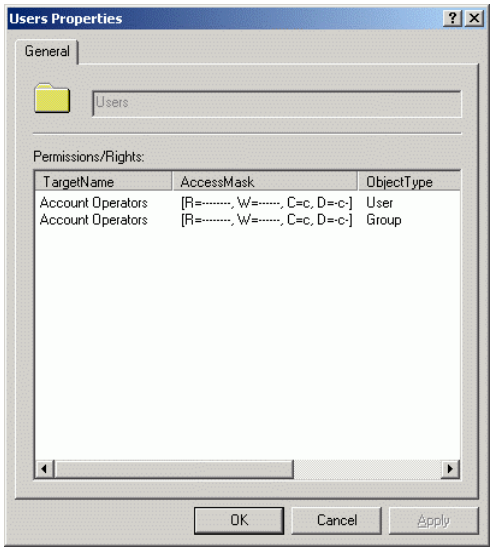


Abbildung 4.5: Eigenschaftenfenster des Users-Containers

Dabei werden die einzelnen Berechtigungen durch entsprechende Buchstaben abgekürzt (siehe Kapitel 3.7.4, Tabelle 3.15) und einer der aus der komprimierten Rechtedarstellung bereits bekannten Kategorien zugeordnet. Aus Abbildung 4.5 geht eindeutig hervor, dass die Gruppe der „Account Operators“ – wie zuvor bereits erwähnt – dazu berechtigt ist, User- und Group-Objekte im Users-Container anzulegen und zu löschen.

5 Referenzen

5.1 Literatur

[Ach] Achleitner, M.: „Analyse der Windows 2000 Rechtestruktur in NTFS und Registry“; Diplomarbeit; Johannes Kepler Universität Linz; (erscheint voraussichtlich 3Q 2002)

[Den] Denning, D.E.: „Information Warfare and Security“; Addison Wesley 1999; ISBN: 0-201-43303-6

[Han] Hanner, K.: „Analyse und Archivierung von Benutzerberechtigungen in einem Windows NT Netzwerk“; Diplomarbeit; Johannes Kepler Universität Linz; 1998

[Hel] Helml, T.: „Darstellung von Rechtestrukturen in Verzeichnisdiensten (am Beispiel Active Directory)“; Diplomarbeit; Johannes Kepler Universität Linz; 2000

[Hoe1] Hörmanseder, R., Hanner, K.: „Managing Windows NT file system permissions (A security tool to master the complexity of Microsoft Windows NT file system permissions)“; Academic Press; Journal of Network and Computer Applications (1999) 22, Seite 119 ff.

[Ise] Iseminger, D.: „Active Directory Services for Microsoft Windows 2000, Technical Reference“; Microsoft Press 2000; ISBN: 0-7356-0624-2

[Mmc1] Microsoft Press: „Microsoft Management Console Design and Development Kit“; Microsoft Press 2000; ISBN: 0-7356-1038-X

[Msc1] Microsoft Press: „Microsoft Windows 2000 Server: Verteilte Systeme. Die technische Referenz“; Microsoft Press 2000; ISBN: 3-86063-273-6

[Msc2] Microsoft Press: „Microsoft Windows User Experience“; Microsoft Press 1999; ISBN: 0-7356-0566-1

[Ott] Ottman, T.; Widmayer P.: „Algorithmen und Datenstrukturen“; 2. Auflage; BI-Wissenschaftsverlag; ISBN: 3-411-16602-9

[Rom] Romano, T.: „A (short) History of MMC“, Microsoft Management Console Design and Developers Kit, Seite 3 f.; Microsoft Press 2000, ISBN: 0-7356-1038-X

[Sch] Schmitzberger, H.: „SAT2 Controller Snap-in“; Dokumentation; Johannes Kepler Universität Linz; (erscheint voraussichtlich 2Q 2002)

5.2 Internet

[Hoe1] Hörmanseder, R., Hanner, K.: „Managing Windows NT file system permissions (A security tool to master the complexity of Microsoft Windows NT file system permissions)“
http://www.fim.uni-linz.ac.at/sat/sat_description.htm

[Hoe2] Hörmanseder, R.: „A user centered view of security for the NT file system, Registry and Active Directory (SAT Prototype + Future work & Future Plans)”, November 2000
http://www.fim.uni-linz.ac.at/sat/sat_plans.pdf

[Hel] Helml, T.: „Darstellung von Rechtestrukturen in Verzeichnisdiensten (am Beispiel Active Directory)“; Diplomarbeit; Johannes Kepler Universität Linz; 2000
http://www.fim.uni-linz.ac.at/sat

[Mmc2] Microsoft Platform SDK – Microsoft Management Console Documentation
http://msdn.microsoft.com/library/en-us/mmc/mmcstart_1dph.asp

[Mmc3] Microsoft Platform SDK – MMC Technical Articles
http://msdn.microsoft.com/library/en-us/dnmmc/html/msdn_mmcfaq.asp

[Msdn] Microsoft MSDN Library Visual Studio 6.0, Visual C++ Documentation
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcedit98/HTML/vcstartpage.asp

[Mskb] Microsoft Knowledge Base: „Management Console Support Center“
http://support.microsoft.com/default.aspx?scid=fh;EN-US;MMC

[Msng] Microsoft MMC Newsgroup
nntp://news.microsoft.com/microsoft.public.management.mmc

[Msng1] Microsoft MMC Newsgroup Arcticle; MeanGene (MS-Support): „RE: MS Recommends don't use ATL Wizard?”; 06.02.2002
nntp://news.microsoft.com/microsoft.public.management.mmc

[Msng2] Microsoft MMC Newsgroup Article; MeanGene (MS Support): „RE: MMC Book?”; 01.02.2002
nntp://news.microsoft.com/microsoft.public.management.mmc

[Msr] MSR - Microsoft Research, Cambridge, UK
http://www.research.microsoft.com/labs/cam.asp

[Sat] Projekt: Security Analysis Tool (SAT), Institut für Informationsverarbeitung und Mikroprozessortechnik, Johannes Kepler Universität Linz
http://www.fim.uni-linz.ac.at/sat

6 Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

.....

Zarda Gerald

7 Lebenslauf

➢ Persönliche Daten

Name	Gerald Zarda
Geboren am	23.07.1974 in Ried im Innkreis
Nationalität	Österreich
Religion	röm.kath.
Familienstand	ledig
Adresse	Burgergarten 1 4971 Aurolzmünster
Telefon	(+43) 0664/3448046
eMail	gerald.zarda@gmx.net
Eltern	Franz Zarda, geb. 1952, Contract & Procurement Manager, Fa. Puerstinger Asia Pacific, Singapore Ingrid Zarda, geb. 1954, Sekretärin, Berufschule 1, Ried i. I.
Geschwister	Wolfgang Zarda, geb. 1977, Angestellter, Fa. Eurol, Ried i. I.

➢ Schulbildung

1980 - 1984	Volksschule, Aurolzmünster
1984 - 1989	Bundesgymnasium, Ried im Innkreis
1989 - 1993	Bundesoberstufenrealgymnasium, Schwerpunkt: Informatik, Ried i. I. Abschluss: Matura (Wahlfächer: Informatik, Englisch, Philosophie)

➢ Studium

1994 - 2002	Diplomstudium Informatik, Wahlfachgruppe: "Network Security", Johannes Kepler Universität Linz
1999	Entwicklung einer "Allgemeinen Compiler Schnittstelle" für POW! (Programmer´s Open Workbench), Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)
2001	Mitarbeit am Projekt: SAT 2 (Security Analysis Tool 2), Entwicklung eines Snap-in für die Microsoft Management Console (MMC) zur Visualisierung von Rechtestrukturen, Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)
2001 - 2002	Erstellen der Diplomarbeit "Visualisierung von Rechtestrukturen in Windows 2000 Netzwerken“ im Rahmen des SAT 2 - Projektes