JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Datenauswertung der Rechtestrukturen unter Windows 2000 und deren Visualisierung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

DIPLOMINGENIEUR

in der Studienrichtung

INFORMATIK

Angefertigt am *Institut für Informationsverarbeitung und Mikroprozessortechnik*

Betreuung:

o. Prof. Dr. Jörg R. Mühlbacher

Dipl.-Ing. Rudolf Hörmanseder

Eingereicht von:

Heinrich Schmitzberger

Linz, *November 2004*

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die zur Erstellung dieser Diplomarbeit beigetragen haben.

In erster Linie möchte ich meinem Betreuer o. Prof. Dr. Jörg R. Mühlbacher, dem Vorstand des Instituts für Informationsverarbeitung und Mikroprozessortechnik (FIM), für das mir entgegengebrachte Vertrauen und für die Bereitstellung der benötigten Laborressourcen den Dank aussprechen.

Mein Dank gilt weiters Herrn Dipl.-Ing. Rudolf Hörmanseder, dem Initiator des ***Security Analysis Tool 2.0*** Projekts. Er stand mir stets mit Rat und Tat zur Seite und hat es verstanden, mir die für das Projekt nötige Motivation zu vermitteln.

Des Weiteren danke ich der MSR – Microsoft Research Cambridge (UK) für die finanzielle Unterstützung des ***Security Analysis Tool 2.0*** Projekts.

Besonderer Dank gilt nicht zuletzt meiner Familie, allen voran meiner Mutter Christine und meinem Vater Heinrich, die mir meine Ausbildung ermöglicht und mich stets in all meinen Entscheidungen unterstützt haben.

Inhaltsverzeichnis

1 Das Security Analysis Tool

1.1 Motivation

„Sicherheit ist ein breites Thema. In der einfachsten Form soll sichergestellt werden, dass neugierige Leute keine Nachrichten anderer lesen oder ändern können. Auf dieser Ebene hat Sicherheit die Aufgabe, unberechtigte Leute davon abzuhalten, auf entfernte Dienste zuzugreifen. … Die meisten Sicherheitsprobleme werden absichtlich und böswillig verursacht. … Netzwerksicherheit verlangt viel mehr, als das Netz von Programmierfehlern freizuhalten."

[Tan00/S.613]

Dieses Zitat soll die Brisanz des Themas Sicherheit in Computernetzen auf abstrakte Weise verdeutlichen. Tatsächlich wird diese Thematik jedoch häufig unterschätzt. Wie aus etlichen Berichten und Statistiken hervorgeht, ist die Hauptursache für Sicherheitsprobleme im internen LAN falsche oder unzureichende Administration. Das Thema Sicherheit erhält in diesem Zusammenhang üblicherweise erst einen angemessenen Stellenwert, wenn bereits Schaden entstanden ist. Dabei sollte das Motto vielmehr „Prävention statt Schadensbegrenzung“ lauten. Um zu gewährleisten, dass sensible Daten nicht in *„falsche Hände*“ geraten, sollte Netzwerksicherheit ein wesentlicher Aspekt in der Planung und Instandhaltung eines jeden Computernetzes sein.

Technisch betrachtet existiert der Terminus Netzwerksicherheit als Sammelbegriff für eine Vielzahl von Paradigmen, Richtlinien und Programmen, deren gemeinsamer Sinn in der Absicherung der Computernetzwerkinfrastruktur und den damit verbundenen Ressourcen vor unerlaubtem Zugriff besteht. Um Netzwerksicherheit aufrecht zu erhalten, gibt es für heutige Administratoren eine Reihe von relevanten Themen:

- ***Antivirenprogramme***
- ***Firewalls***
- ***Kryptografie***
- ***Protokollierungsprogramme***
- ***Netzwerkanalyseprogramme***
- ***Zugriffskontrolle mittels Benutzerrechten***

Jedes dieser Themen beschreibt einen anderen Aspekt der Netzwerksicherheit und umfasst im Zuge dessen eine Ansammlung von Problemen, Lösungsansätzen und Implementierungen. Betrachtet man die daraus resultierende Vielzahl an verschiedenen Problemfeldern, so verdeutlicht dies den Schluss, dass es keine triviale Aufgabe ist, für die Sicherheit eines Netzwerks Sorge zu tragen.

Laut der Top 14 der allgemeinen Sicherheitslücken in Computernetzwerken, wie sie in [McCl03/S.815] präsentiert werden, besteht ein erhebliches Gefahrenpotenzial in den Benutzerrechten der LAN-Nutzerkonten. Auf Platz 6 wird als Bedrohung angeführt, dass in einem System Benutzer- und Testkonten mit überflüssigen Privilegien existieren. Platz 10 dieser Top 14 hält die zu freizügige Datei- und Verzeichniszugriffsrechtvergabe. Findige Angreifer wissen diese „Konfigurationsschwächen" durchaus auszunutzen, um an Daten zu gelangen, die nicht für ihre Augen bestimmt sind.

Um dieser Art von Angriffen vorzubeugen, bedarf es einer konsistenten Administration der Benutzerberechtigungen. Moderne Betriebssysteme bieten hierfür mittlerweile einen breiten Funktionsumfang an Zugangsbeschränkungsmöglichkeiten und Administrationswerkzeugen, die jedoch die Verwaltung eines Netzwerks nicht immer gerade erleichtern. Dazu kommt noch, dass die Komplexität der Zugriffskontrolle proportional zur Anzahl der Benutzer und zur Größe des Netzwerks steigt.

Die Intention des ***SAT*** Projekts ist es, Administratoren von Systemen der *Microsoft Windows NT/2000/XP/2003* Produktreihe bei der komplexen Aufgabe der Benutzerverwaltung und Zugriffskontrolle zu unterstützen. Das ***Security Analysis Tool (SAT)*** stellt ein Werkzeug dar, das dem Administrator eine Sicht auf das Netzwerk und dessen Ressourcen bietet, die in bisherigen Werkzeugen fehlt. Es wird versucht, dem Administrator die Konsistenz seiner Rechteverwaltung aufzuzeigen und auf eventuelle Sicherheitslöcher hinzuweisen.

An dieser Stelle sei erwähnt, dass sich das ***SAT*** Projekt bisweilen ausschließlich mit proprietären Technologien des Betriebssystemproduzenten Microsoft wie dem NT File System, dem Active Directory und der Microsoft Management Console befasst. Eine Erweiterung des Projekts auf andere gängige Betriebssysteme ist im Prinzip möglich.

1.2 Übersicht

Diese Diplomarbeit befasst sich mit der Analyse und Visualisierung der Rechtestrukturen in *Microsoft Windows NT/2000/XP/2003* Netzwerken, wobei zum einen NTFS 5 und zum anderen das Active Directory Service (ADS) in Betracht gezogen werden. Zu diesem Zweck und als Hauptgegenstand dieser Arbeit wurde das ***Security Analysis Tool (SAT) 2.0*** implementiert und ist zum Zeitpunkt der Veröffentlichung dieser Arbeit im Netz unter [Sat04] frei verfügbar.

Dieses erste Kapitel gibt eine allgemeine Einführung in die Thematik der Netzwerkadministration mit besonderem Bezug auf die Absicherung eines in Betrieb befindlichen LANs durch korrekte Benutzerverwaltung. Im Zuge dessen wird die Grundidee des ***SAT*** Projekts erörtert und der Werdegang des ***Security Analysis Tools*** beschrieben.

Kapitel 2 versucht, ***SAT 2.0*** im Detail zu beschreiben. Dazu werden die grundsätzlichen Überlegungen und Anforderungen, die an ein Sicherheitstool für *Microsoft* Netzwerke gestellt werden, umrissen und erläutert. Es wird auf die Funktionalität von ***SAT 2.0*** eingegangen und die Architektur des Tools beschrieben. Des Weiteren wird geschildert, welche Voraussetzungen zum Betrieb des ***SAT*** von Nöten sind und wie es zu bedienen ist. Zum Abschluss dieses Kapitels werden verschiedene Beispielszenarien erläutert, die die Funktionsweise des Tools verbildlichen sollen.

Im dritten Kapitel wird auf das Rechtesystem von *Microsoft Windows 2000* eingegangen, das die Basis zum Verständnis von ***SAT 2.0*** darstellt. Dazu werden die Algorithmen und Datenstrukturen, die für die Sicherheit im NTFS 5 und im ADS verantwortlich sind, aufgeschlüsselt und im Detail erläutert. Daraus resultiert schließlich das Datenbankschema, das im letzten Abschnitt dieses Kapitels behandelt wird.

Kapitel 4 beschreibt die Microsoft Management Console (MMC) und stellt eine Einführung in die Programmierung dieser mithilfe der Active Template Library (ATL) 3.0 dar. Da es in diesem Bereich an Literatur mangelt – das einzige, bis zur Fertigstellung dieser Arbeit verfügbare Werk für MMC Programmierung [Mmc00] nimmt leider keinen Bezug auf die ATL Programmierung und deren Vorteile –, hat sich der Autor auch vorgenommen, die bei der Implementierung der MMC Komponenten von ***SAT 2.0*** gesammelten Erfahrungen zusammenzufassen, um eine Anleitung für eventuelle Weiterentwicklungen in diesem Bereich zu geben.

1.3 Grundzüge der angewandten Netzwerkadministration

Die Administration eines in Betrieb befindlichen LANs ist eine Aufgabe, die sich aufgrund der Vielfalt der zu berücksichtigenden Aufgabenfelder als nicht trivial darstellt. Einer der wesentlichsten Aspekte, wenn nicht der wesentlichste überhaupt, ist die Verwaltung des Zugriffs auf die Ressourcen des Netzwerks. Dieser Zugriff wird durch Benutzerberechtigungen geregelt und kann für einzelne Benutzer wie auch für Benutzergruppen erlaubt oder explizit untersagt werden. Somit hat der Administrator des Netzes die Möglichkeit, die Komponenten seines LANs durch korrekte Rechtevergabe zu schützen.

Ein Problem, das aus diesem Mechanismus erwächst, ist die **objektzentrierte Sicht** des Netzwerks und seiner Ressourcen. Da die Rechte in modernen Serverbetriebssystemen immer nur auf ein Objekt bzw. auf eine Gruppe von Objekten vergeben werden können, bauen die gebräuchlichen Administrationsanwendungen auf diesem Konzept auf und bieten dementsprechend lediglich diese Betrachtungsweise an. Um herauszufinden, auf welchen Dateien Benutzer X Zugriff hat, müsste man theoretisch jede Datei einzeln betrachten und die Sicherheitsinformationen aufsummieren. Man stelle sich nun ein durchschnittliches Firmennetzwerk mit einem einzelnen Server vor, der unter anderem als File Server und Web Server dient. Geht man von einer systematisch gegliederten Verzeichnisstruktur aus, so sind im Allgemeinen auf diesem Server folgende Typen von Verzeichnissen zu finden:

- **Benutzerverzeichnisse:** In diesen Verzeichnissen werden persönliche Dateien der jeweiligen Benutzer abgelegt. Zugriff hat nur der Besitzer des Verzeichnisses selbst. Für die restlichen Systembenutzer sollte der Zugriff abgelehnt werden.

- **Programm- und Installationsverzeichnisse:** … enthalten in der Regel die Software, die in der Firma verwendet wird, somit müssen auf diesem Verzeichnis sämtliche Systembenutzer Zugriff haben.

- **Projektbezogene Verzeichnisse:** Diese Verzeichnisse enthalten Daten zu bestimmten Projekten. Der Zugriff ist folglich nur einer Benutzergruppe erlaubt, die an dem jeweiligen Projekt arbeitet. Anderen Benutzern ist der Zugriff zu untersagen.

- **Öffentliche Verzeichnisse:** In diesem Verzeichnistyp werden Daten gespeichert, die über einen Dienst (HTTP Server, FTP Server, …) im Internet angeboten werden. Diese Daten sind im Allgemeinen für anonymen (Lese-)Zugriff bestimmt, dürfen aber nur von bestimmten Benutzern geschrieben bzw. geändert werden.

Daraus ergibt sich eine Baumstruktur von Zugriffsberechtigungen, die zusätzlich noch durch eventuelle Vererbungsregeln verkompliziert werden kann, wie es in *Microsoft Windows NT/2000/XP/2003* Betriebssystemen Usus ist. Bedenkt man nun die Anzahl an verschiedenen Benutzern, die in einem durchschnittlichen Netzwerk existieren, so kann man sich leicht vor Augen führen, dass die Konsistenz der Rechtestruktur in diesem exemplarischen Firmennetz mit objektzentrierten Werkzeugen schwer überprüfbar ist.

Diese Überlegungen resultierten schließlich in der Idee des ***Security Analysis Tools***, einem Werkzeug, das versucht, dem Administrator einen detaillierten Überblick über die Rechteverteilung in seinem Netzwerk zu geben und somit die Mängel in den bisherigen Administrationswerkzeugen heutiger Betriebssysteme aufbessert. Dazu bedient sich ***SAT*** einer **benutzerzentrierten Sicht** auf das Netz, d.h., es besteht die Möglichkeit, für jeden beliebigen Benutzer oder jede beliebige Benutzergruppe eine Auflistung aller Objekte im Netzwerk zu erstellen, auf die ein Zugriffsrecht besteht. Dadurch stellt ***SAT*** ein Netzwerkanalysewerkzeug dar, das Administratoren bei ihrer Arbeit unterstützt und die Netzwerksicherheit steigert.

1.4 Zur Geschichte von SAT

Die ursprüngliche Idee zum ***SAT*** Projekt entstand im Jahr 1995 am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) der Johannes Kepler Universität. Der Institutsmitarbeiter und Netzwerkadministrator Hr. Dipl.-Ing. Rudolf Hörmanseder bemerkte funktionale Mängel bezüglich der Benutzerverwaltung und Zugriffsdarstellung im zu dieser Zeit am FIM eingesetzten Netzwerk-Betriebssystem *Microsoft Windows NT 3.1* und entwickelte ein Konzept für ein Werkzeug, das diese Mängel ausbügeln sollte und in [Hör99] beschrieben wird. Dies war die Geburtsstunde des ***Security Analysis Tools***. Zusammen mit dem damaligen Studenten Kurt Hanner wurde dieses Konzept implementiert und mit Anfang 1999, als ***SAT 1.0 Beta*** veröffentlicht. Diese Version beschränkte sich auf die Analyse des Dateisystems NTFS 4. Nähere Details zu diesem System und zur Implementierung sind in [Han98] nachzulesen.

Im Februar 2000 zeichnete sich mit der Markteinführung von *Microsoft Windows 2000* ein Problem für das ***SAT*** ab. Das neue *Microsoft* Betriebssystem enthielt eine Vielzahl an Neuerungen, unter anderem auch das Dateisystem NTFS 5, das modernere Sicherheitskonzepte implementierte und leider nicht mehr mit ***SAT 1.0*** kompatibel war. Zusätzlich wurden noch weitere Besonderheiten im Bereich Netzwerkadministration vorgestellt. Das Active Directory Service wurde eingeführt, das die Firmenstruktur direkt auf das Netzwerk abbilden und den unternehmensweiten Datenaustausch vereinfachen sollte. Darüber hinaus wurde ein Versuch gestartet, der Computeradministration unter *Windows* ein einheitliches Gesicht zu verleihen, um die Bedienung der mittlerweile zahlreichen Administrationswerkzeuge überschaubarer zu gestalten. Dies wurde durch die Microsoft Management Console (MMC) angestrebt, die als eine Art Einheitsoberfläche fungiert und eine beliebige Anzahl an Tools verwalten kann.

Diese Neuerungen waren ausschlaggebend für die Weiterentwicklung des Sicherheitsanalyse-Tools. In Zusammenarbeit mit den Studenten Thomas Helml, Gerald Zarda und Michael Achleitner konzipierte Herr Hörmanseder das Projekt ***SAT 2.0***, das sich die Integration der neuen Administrations- und Sicherheitskonzepte von *Windows* in die ***SAT***-Analyse zum Ziel setzte (vgl. dazu [Hör00]).

In diesem Projekt wurde Herr Helml mit der Aufgabe betraut, den Active Directory Scanner zu entwickeln. Zusätzlich fiel die Neuimplementierung der Datenbankschnittstelle des ***SAT2*** Systems in seine Zuständigkeit. Die Ergebnisse seiner Arbeit sind unter [Hel00] nachzulesen.

Der Computer-, Gruppen- und Benutzerscanner wurde von Herrn Zarda implementiert. Im Weiteren wurde er mit der Aufgabe betraut, die Visualisierungskomponente zu entwickeln. Da das Projektteam zum Zeitpunkt der Arbeit von Herrn Zarda noch über keinerlei Erfahrungen in der MMC Programmierung verfügte, wurde lediglich ein erster Prototyp erstellt. Seine Ergebnisse stehen unter [Zar02] zur Verfügung.

Der Zuständigkeitsbereich von Herrn Achleitner war die Umsetzung des NTFS 5 Scanners sowie des Registry Scanners, der jedoch nicht mehr fertig gestellt wurde. Seine Arbeiten in diesem Bereich werden in [Ach02] beschrieben.

Der Autor dieser Arbeit begann seine Teilnahme am ***SAT2*** Projekt Anfang des Jahres 2002 mit der Umsetzung der Controller Applikation. Aufgrund seiner Erfahrung mit der Microsoft Management Console wurde er mit der Neuimplementierung der Visualisierungskomponente betraut, die den praktischen Aspekt dieser Diplomarbeit darstellt.

Mit dem Abschluss dieser Diplomarbeit wird zugleich die Sicherheitsanwendung ***Security Analysis Tool 2.0*** veröffentlicht, die das Resultat des ***SAT2*** Projekts darstellt. In dieser neu überarbeiteten Version wird dem Administrator eines *Microsoft Windows NT/2000/XP/2003* Netzwerks nunmehr die Möglichkeit geboten, sowohl NTFS als auch ADS Analysen durchzuführen. Sowohl die Steuerung als auch die Ergebnisse der Analysen erscheinen als so genannte Snap-Ins für die MMC und bedienen sich eines *Microsoft SQL Servers* als Backend für die Datenarchivierung.

2 SAT 2.0 – Die nächste Generation

Dieses Kapitel beschreibt die grundsätzlichen Überlegungen, die hinter dem ***SAT 2.0*** System stehen und wie es anzuwenden ist. Es sollte von all denen gelesen werden, die sich ein Bild über die Funktionalität von ***SAT 2.0*** machen wollen bzw. sich fragen, aus welchem Grund sie gerade dieses System in ihrem eigenen Netzwerk verwenden sollten.

Dem Leser dieser Arbeit soll an dieser Stelle nochmals bewusst gemacht werden, dass sich das ***SAT 2.0*** System bisweilen ausschließlich auf die *Microsoft NT* Produktreihe bezieht. Eine Entwicklung der ***SAT*** Idee für andere Betriebssysteme ist denkbar, ist jedoch nicht Teil dieser Diplomarbeit.

2.1 Anforderungen an das Security Analysis Tool

„… Und nicht zuletzt bleibt die NT-Familie deswegen im Visier der Hacker, weil die Plattform standardmäßig eine Vielfalt an Features und Funktionen aktiviert. … Eine der wichtigsten Regeln der Sicherheit lautet: Das Sicherheitsrisiko eines Systems steigt proportional zu dessen Komplexität – und aus den Sünden der Vergangenheit, der Aktivierung der maximalen Funktionalität bei der Default-Installation, hat Microsoft nichts gelernt.“

[McCl03/S.194 ff]

Es ist eine Tatsache, dass die Firma *Microsoft* in ihrer Betriebssystemproduktreihe mit einer geraumen Anzahl von Sicherheitsproblemen zu kämpfen hatte und immer noch hat. Dabei stehen ohne Zweifel zwei Fakten im Vordergrund. Zum einen trägt die unsichere Standardkonfiguration wesentlich zur Anzahl der Sicherheitslöcher bei, zum anderen ist es nicht von der Hand zu weisen, dass sich die *Microsoft Windows NT* Produktreihe einer großen Beliebtheit und folglich einer weiten Verbreitung erfreut.

Die Beliebtheit der *Microsoft* Systeme zeichnet sich als ein zweischneidiges Schwert ab. Auf der einen Seite profitiert jeder Administrator von den Vorteilen eines Systems, dessen Akzeptanz im Endbenutzerbereich auf einer jahrelangen Marktführerposition aufbaut. Die Support-Infrastruktur, die von *Microsoft* sowohl im Entwickler- als auch im Anwenderbereich etabliert wurde, sucht heutzutage ihres gleichen. Andererseits lockt gerade diese Beliebtheit immer mehr eifrige Hacker an, denen es Genugtuung bereitet, neue Schwachstellen aufzuspüren und diese auch auszunutzen. Von diesen Hackern wird es als Prestigefrage betrachtet, den „Goliath“ der Branche in die Knie zu zwingen, was nicht zuletzt auf die in manchen Kritikerkreisen umstrittene Firmenpolitik des Redmonder Softwareriesen zurückzuführen ist. Dies hat zur Folge, dass sich Rechnernetzbetreiber weltweit immer wieder mit neuen Bedrohungen aus dem Internet konfrontiert sehen.

Die unsichere Standardkonfiguration, in der die einzelnen Mitglieder der *Windows NT* Produktreihe ausgeliefert werden, ist zu einem großen Teil für diese Sicherheitslöcher, mit denen Computeranwender weltweit täglich konfrontiert werden, verantwortlich. So ist es zum Beispiel ein bekanntes Administrationsproblem gewesen, dass die *Windows 2000* Standardvererbung der Benutzergruppe „Jeder“ die vollen Zugriffsrechte für einen neu angelegten Ordner zugewiesen hat. Dieses Problem hat in Verbindung mit der standardmäßigen Installation und Aktivierung der *Internet Information Services (IIS)* etlichen Internetbenutzern unautorisierten Zugriff auf vertrauliche Daten verschafft.

Daraus drängt sich ein deutliches Bild auf: Um die Sicherheit eines Systems gewährleisten zu können, sollte jeder Administrator genau darüber Bescheid wissen, auf welche Daten welcher Benutzer welche Rechte besitzt. Als Verantwortlicher sollte man sich nicht darauf verlassen, dass die Default Benutzerrechte vom Betriebssystem sicher gesetzt werden.

Das ***Security Analysis Tool*** in seiner aktuellen Version 2 versucht, auf Administrationsprobleme der *Windows NT* Betriebssystemfamilie Antwort zu geben, indem es systematisch die Objekte und Ressourcen im *Windows* Netzwerk analysiert, aufschlüsselt, und Mängel in der Konfiguration herausstreicht. Dabei bietet das ***SAT*** System eine Lösung an, die in dieser Form zum Zeitpunkt der Erstellung dieser Arbeit noch nicht am Softwaremarkt erhältlich war und deren Vorteile im Folgenden näher erläutert werden.

Ein bekanntes Problem in der Administration von *Windows* Netzwerken zeichnet sich durch die objektzentrierte Sicht der Tools und Verwaltungsanwendungen ab, die im Allgemeinen Verwendung finden. Unter objektzentrierter Sicht hat man sich die Betrachtungsweise vorzustellen, die dem Anwender in modernen Betriebssystemen üblicherweise geboten wird. Es ist gebräuchlich, dass Zugriffsberechtigungen immer nur auf ein Objekt bzw. in weiterer Form auf eine Gruppe von Objekten vergeben werden. Als Objekte werden in diesem

Zusammenhang sowohl Ressourcen wie Netzwerkdrucker oder Backupvorrichtungen wie auch Dateisystemobjekte - Ordner, Dateien, Active Directory Objekte, etc. - angesehen. Dieses einfache Schema zieht sich durch die gesamte Palette der Administrationsapplikationen, nicht zuletzt deshalb, weil es die betriebssysteminterne Sicht widerspiegelt. Um beispielsweise der Frage auf den Grund zu gehen, ob bzw. auf welche Objekte der Benutzer X auf dem Netzlaufwerk Y Zugriffsrechte besitzt, müsste man mit herkömmlichen Anwendungen theoretisch sämtliche Dateien und Ordner betrachten. Dass dies einen unzumutbaren Zeitaufwand bedeutet ist evident.

Diese Problematik stellt zugleich die Hauptanforderung an das ***SAT*** System dar. ***SAT2*** bietet seinem Anwender eine benutzerzentrierte Sicht auf das Netz mit sämtlichen darunter befindlichen Objekten und bildet diese als Baum ab. Dabei werden die Zugriffsrechte des analysierten Benutzers auf NTFS und ADS Objekte durch farblich codierte Symbole repräsentiert. Die Aufschlüsselung dieser Zugriffsrechte wird im Kapitel 3.2 ausführlicher beschrieben.

Ein essentieller Punkt in dieser benutzerzentrierten Analyse der Zugriffsberechtigungen ist, dem Anwender eine benutzerfreundliche Applikation zu präsentieren. Zu diesem Zweck wurde als Framework für das ***SAT2*** Projekt die Microsoft Management Console gewählt, die von *Microsoft* mit eben jener Intention entwickelt wurde, Administrationsaufgaben zu übernehmen und zu vereinfachen (vgl. dazu Kapitel 4.1). Die aus dem Dateiexplorer bekannte Baumdarstellung der Objekte ist zweckmäßig für die Ziele des ***SAT*** und wurde daher adaptiert. Analog zur tatsächlichen Dateisystemstruktur wird ein Baum aus Dateisystemobjekten aufgebaut und ist - je nach Analysemodus - darauf ausgerichtet, dem Administrator eine intuitive Navigation zu den Problemstellen bzw. Sicherheitslücken zu bieten.

Um den Grad eines Sicherheitsrisikos zu verdeutlichen und vor allem zu visualisieren, wird eine Art Ampelstrategie verfolgt. Dabei resultieren die Rechte des analysierten Benutzers in einer nach der Verkehrsampel geordneten Einfärbung des jeweiligen Objekts, wobei verständlicherweise die Farbe rot auf volle Zugriffsrechte hinweist und vermeintlichen Handlungsbedarf signalisieren soll. Eine detaillierte Erläuterung der verschiedenen Analysen und Objekteinfärbungen ist in Kapitel 2.4.2 zu finden.

Weitere Anforderungen an das ***SAT 2.0*** System betreffen den Arbeitskomfort bzw. die Performanz der Applikation. Das System wurde bewusst auf dezentrale Anwendbarkeit hin ausgelegt. Es muss dem Administrator möglich sein, sowohl den Scanvorgang der einzelnen Rechner als auch die Analyse der Ergebnisse von jedem beliebigen Terminal im Netz aus zu starten bzw. durchzuführen.

Der erwähnte Scanvorgang, der im allgemeinen Sinne als Datensammlungsprozess zu verstehen ist, ist die zeitaufwendigste Routine des ***SAT*** Prozesses. Dieser kann auch, falls es die Anzahl der zu scannenden Objekte erfordern würde, über Nacht bzw. übers Wochenende durchgeführt werden, da die Scannerkomponente (siehe Kapitel 2.2) als Dienst implementiert wurde.

Aufgrund der im Laufe der Entwicklung des ***SAT2*** Prototypen aufgetauchten Performanzprobleme - sowohl bei der Analyse als auch bei der Datensammlung -, wurde ein Auge auf algorithmenorientierte und datenbankgerichtete Geschwindigkeitsoptimierung gerichtet. Ein Feature, das bereits in der Vorgängerversion Einzug in die Anforderungsliste fand, ist die Möglichkeit, der Scannerkomponente verschiedene Kompressionsstufen einzustellen. In der aktuellen ***SAT*** Version werden mehrere Kompressionsstufen (siehe Kapitel 2.4.3) angeboten, um selbst für größere Netzwerke applikabel und stabil zu bleiben. Des Weiteren wurden die Abfrageroutinen des Datenbankinterfaces in der Scannerkomponente einer Überarbeitung und Optimierung unterzogen. Um den Baumaufbau in der MMC fließender zu präsentieren und die Wartezeiten bei der Analyse so gering wie möglich zu halten, wurde Wert auf die Analyseabfragen gegen die Datenbank gelegt. Diese bilden einen wesentlichen Bestandteil der Viewer Komponente und wurden gegenüber dem Prototyp verbessert und auf Aufbaugeschwindigkeit in der Management Console getrimmt.

Als Anforderung hinsichtlich einer Weiterentwicklung des Systems wurde eine automatische Komponentenupdatefunktion implementiert. Diese Funktion stellt sicher, dass sich die Scannerkomponente des ***SAT*** vor jedem Scanprozess auf dem entsprechenden Netzcomputer immer auf dem aktuellsten Stand befindet. Es muss lediglich dafür gesorgt werden, dass eine neue Version auf dem Controller-Rechner verfügbar ist (siehe Kapitel 2.4.1). Somit sind Produktupdates über das gesamte Netzwerk von zentraler Stelle aus durchführbar.

2.2 Architektur

Das ***Security Analysis Tool 2*** besteht aus drei Komponenten. Diese Komponenten arbeiten nach dem Master-Slave-Prinzip, das laut [Bus98/S.245] so umrissen werden kann:

„Das Master-Slave-Muster unterstützt den Entwurf von Systemen, die fehlertolerant sind, Parallelverarbeitung nutzen oder Rechengenauigkeit bieten. Ein Meister (engl. master) teilt die Arbeit zwischen identischen Gehilfen (engl. slave) auf und ermittelt aus den Ergebnissen, die die Gehilfen liefern, das Endergebnis.“

SAT2 bedient sich dieses Prinzips, um die parallele Datensammlung, die auf allen Netzwerkrechnern wenn nötig gleichzeitig durchgeführt werden kann, gewährleisten zu können. Dabei kann man die Komponenten (siehe auch Abbildung 2.2.1) wie folgt einteilen:

- **Master Komponente - Controller**
  Der Controller bildet die Steuereinheit des Systems und sorgt dafür, dass die eingesetzte Scannerversion am aktuellsten Stand bleibt. Weiters ist er auch noch für die Versionierung der Datenbank verantwortlich.

- **Master Komponente - Viewer**
  Die Viewer Komponente ist für die Datenanalyse zuständig. Sie kann als Frontend von ***SAT2*** verstanden werden und dient zur Visualisierung von entdeckten Sicherheitsmängeln. Ferner gibt sie Überblick über die Benutzer- und Gruppenverteilung im Netzwerk und organisiert zu Vergleichszwecken die einzelnen durchgeführten Analysen.

- **Slave Komponente - Scanner**
  - ***Computer-, Gruppen- und Benutzerscanner (CGU-Scanner)***
    … liest SID Informationen der jeweiligen Security Principals (siehe Kapitel 3.1.1) aus und verwaltet die Gruppenzugehörigkeit. Diese Komponente ist für die Rechteanalyse unumgänglich, da sich sämtliche Daten, die durch die restlichen Scanner geliefert werden, auf diese SID Informationen beziehen.

- ***Active Directory Service Scanner (ADS-Scanner)***
  Diese Scannerkomponente bildet Active Directory Objekte und das Active Directory Schema auf die Datenbank ab.

- ***NTFS-Scanner***
  … ist für das Auslesen der Festplatten, die auf dem NTFS Dateisystem basieren und sich in den einzelnen Netzwerkrechnern befinden, verantwortlich.

- ***Registry-Scanner***
  Diese Komponente wurde dazu erdacht, die Benutzerrechte auf sämtliche Registry Schlüssel auszulesen. Bisweilen wurde sie noch nicht implementiert.

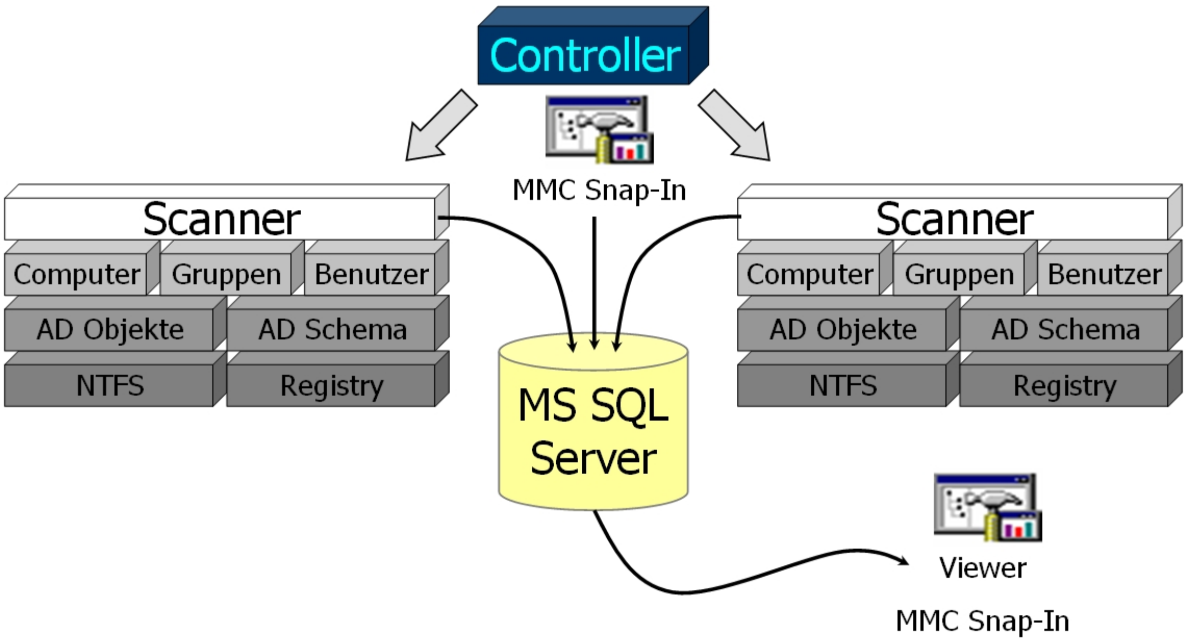


Abbildung 2.2.1: Aufbau der SAT2 Komponenten

Wie aus Abbildung 2.2.1 ersichtlich wird, benutzt das ***SAT2*** System einen Datenbankserver als Backend für die Datenspeicherung. Betrachtet man die Anzahl an Einträgen, die bereits von einem einzelnen Scanprozess auf einem Domain Controller Rechner produziert werden, so stellt ein relationales Datenbasis Management System (DBMS), das in der Größe der zu verwaltenden Daten keine theoretische Beschränkung aufweist, die Mindestanforderung dar, um operabel zu bleiben. Es existieren mehrere Gründe für die Integration eines DBMS in die Architektur des ***SAT*** Systems. In erster Linie werden dadurch die parallele Verarbeitung der Scanvorgänge im Netzwerk und die weder zeit- noch ortsgebundene Auswertung der daraus resultierenden Datenbestände ermöglicht. Des Weiteren können die archivierten Scandaten zu Vergleichen herangezogen werden. Dies kann zusätzliche Schwachpunkte in der ständig migrierenden Netzwerkkonfiguration aufzeigen bzw. den Handlungsbedarf in gewissen Problembereichen verdeutlichen.

Die Controller Komponente (siehe Abbildung 2.2.1), die als MMC Snap-In realisiert wurde, übernimmt die Koordination der Scannerkomponente und der Datenbank. Sie hat dafür zu sorgen, dass die aktuellsten Scannerprogrammdateien auf den zu scannenden Rechnern liegen, bevor dort ein Scan initiiert wird. Da der ***SAT*** Scanner als Dienst agiert, muss der Controller auch die Registrierung dieses Dienstes übernehmen. Weiters kommt dem Controller auch die Aufgabe zu, die Datenbank von älteren Scanversionen zu bereinigen, die nicht mehr gebraucht werden. Dies wird bewusst nicht automatisch durchgeführt, sondern kann durch den Administrator initiiert werden. Um die Durchführung der Scanprozesse möglichst flexibel gestalten zu können, werden im Controller die einzelnen Rechner in Scanroutinen organisiert und mit den gewünschten Scanoptionen konfiguriert. Der Controller kann somit als Steuereinheit des ***SAT2*** Systems verstanden werden.

Die Scanner Komponente wurde in mehrere Subkomponenten gegliedert (vgl. Abbildung 2.2.1). Diese Gliederung wurde aufgrund der Tatsache, dass die Entwicklung des ***SAT2*** im Rahmen mehrerer Diplomarbeiten stattfand, durchgeführt. Dabei wurde versucht, jede Subkomponente auf einen bestimmten Aspekt der Netzwerkanalyse hin auszulegen. Der CGU-Scanner bildet in diesem Konzept die einzige Teilkomponente, von deren Daten die Analyse der Daten der restlichen Scanner abhängig ist, und kann somit als Kernkomponente aufgefasst werden. Die weiteren Scankomponenten sind wahlfrei einsetzbar, können nach Belieben und in jedweder Reihenfolge ausgeführt werden und bieten somit ein hohes Maß an Flexibilität.

Die Auswertung der gescannten Datenbestände findet im ***SAT*** Viewer (Abbildung 2.2.1), der Visualisierungskomponente des ***SAT2*** Systems statt. Diese Komponente führt die benutzer-zentrierten Analysen durch und bietet 7 verschiedene Modi der Datenauswertung an. Wie die Controller Komponente ist auch die Viewer Komponente als MMC Snap-In ausgelegt. Ein Vorteil, der aus dieser Entscheidung erwächst, ist in der dem *Windows* Dateiexplorer verwandten Darstellung der Dateistrukturen zu sehen. Der Umgang mit der Viewer Komponente erfolgt somit benutzerfreundlicher und intuitiver, da die Administration der *Windows* Benutzerrechte auch zum großen Teil im Dateiexplorer passiert.

2.3 Voraussetzungen und Installationsanweisungen

In diesem Kapitel werden die nötigen Anforderungen geschildert, um das ***SAT2*** System benutzen zu können. Unter anderem wird Bezug auf die im Projekt verwendete Software genommen. Schließlich wird die Installation des Systems ausführlich geschildert.

Die Hardwareanforderungen des ***SAT2*** Systems sind einfach zu schildern. ***SAT2*** kann auf jedem Rechner ausgeführt werden, der die Mindestanforderungen von *Windows 2000* erfüllt. Die Verwendung eines Netzwerks ist zwar nicht zwingend erforderlich, wird aber aufgrund der Charakteristik des Systems als evident betrachtet.

Die Ausrichtung des ***SAT*** auf *Microsoft* Betriebssysteme wurde schon in vorherigen Kapiteln (Kapitel 1.1 und Einführung zum Kapitel 2) angedeutet. Dies ist zum einen damit zu begründen, dass das Institut für Informationsverarbeitung und Mikroprozessortechnik der Johannes Kepler Universität, an dem die Grundidee für diese Sicherheitsapplikation entstand, die Betriebssystemreihe *Windows NT* als Basis für das lokale Netz verwendet und somit an einem derartigen Werkzeug interessiert ist. Zum anderen besteht aufgrund der Vormachtstellung von *Windows NT* Betriebssystemen am Markt ein breiteres Anwenderpotenzial als bei vergleichbaren Betriebssystemen.

Das ***SAT 2.0*** wurde für den Einsatz in einer *Windows 2000+* Umgebung konzipiert. Dies beinhaltet die zum Zeitpunkt der Erstellung dieser Arbeit am Softwaremarkt erhältlichen Produkte *Windows XP* und *Windows 2003*. Es besteht die Möglichkeit, die Abwärtskompatibilität zu *Windows NT 4 (SP3+)* zu etablieren. Diese Möglichkeit ist im derzeitigen ***SAT2*** angedacht und im Datenbankschema integriert worden. Bisweilen wurde jedoch die Implementierung mangels Ressourcen an Projektmitarbeitern ausgesetzt.

Als Datenbank Backend wird ein relationales Datenbasis Management System (DBMS) vorausgesetzt, das SQL-92 Kompatibilität aufweist und keinen Beschränkungen bezüglich der maximal verwaltbaren Datengröße unterliegt (wie es beispielsweise bei *Microsoft Access* nicht der Fall wäre). Während der Entwicklungsphase wurde vom ***SAT*** Projektteam der *Microsoft SQL Server 2000* verwendet und wird somit aufgrund der bestehenden Erfahrungen empfohlen. Der Zugriff auf das DBMS wurde mittels ActiveX Data Objects (ADO) realisiert (siehe [Hel00/S.76 ff]).

Das ***SAT2*** System selbst ist als .ZIP Datei verfügbar und kann über [Sat04] bezogen werden. Um die Applikation zu installieren, müssen die Programmdateien in ein temporäres Verzeichnis extrahiert werden. Das Programmpaket enthält eine Batch-Datei namens *install.bat*, die dafür sorgt, dass die folgenden 3 Programmdateien im korrekten Verzeichnis abgelegt und installiert werden:

- *SATViewer.dll*
  Die Viewer Komponente, die eine MMC Snap-In Applikation darstellt.

- *SATController.dll*
  Der Controller des Systems, der ebenfalls als MMC Snap-In implementiert wurde.

- *Sat.exe*
  Die Scannerkomponente, die als einzige der drei Programmdateien auf jedem Rechner verfügbar sein muss, auf dem ein Scan durchgeführt werden soll.

Die Sat.exe Datei ist ein *Windows* Dienst und wird vom Controller auf jedem Rechner, auf dem eine Scanroutine initiiert wird, als Dienst registriert. Der Dienst läuft dabei unter dem Konto *SATAdmin*, das über entsprechende Rechte verfügen muss, um die gewünschte Information auslesen zu können. Um den Dienst zu deinstallieren, kann man Sat.exe mit der Option „remove“ aufrufen.

Die beiden COM in-process Serverapplikationen (siehe Kapitel 4.2) SATViewer.dll und SATController.dll müssen als MMC Snap-In registriert werden, wofür die Installations-Batch-Datei zuständig ist. Um ein Snap-In ohne die Batch-Datei zu installieren, muss man den *Windows*-internen Befehl

- regsvr32 *MeinSnapIn.dll*

ausführen. Die Deinstallation der MMC Snap-Ins wird mittels desselben Befehls durchgeführt. Zusätzlich muss dazu noch die Option „/u“ angegeben werden.

Wie schon weiter oben erwähnt ist das Benutzerkonto *SATAdmin* essentiell für das Funktionieren der ***SAT*** Applikationen. Dabei muss dieses Konto als Mitglied der Gruppe *Administratoren* (lokal) oder als Mitglied der Gruppe *Domain Administratoren* (ADS) im Betriebssystem bzw. am Domain Controller angelegt werden, bevor das ***SAT2*** System angewendet werden kann. Dies ist in erster Linie deshalb erforderlich, da das Auslesen der Sicherheitsinformationen ebenfalls einer Berechtigung bedarf. Was die Verbindung zum DBMS anbelangt, wird ebenfalls der Benutzer *SATAdmin* herangezogen. Dieser sollte auch im DBMS über ausreichend Rechte verfügen, um neue Datenbanken anzulegen und die wichtigsten Operationen darauf ausführen zu können. Wird der *Microsoft SQL Server 2000* verwendet, so empfiehlt es sich, den Benutzer *SATAdmin* in die datenbankinterne Gruppe der *System Administratoren* aufzunehmen, da diese alle erforderlichen Berechtigungen aufweist.

Da die Anforderungen an das aktuelle ***SAT*** System zu einer anderen Programmarchitektur und zu einem umfangreicheren Datenbankschema führten, existiert keine Update Routine von ***SAT 1.01beta*** auf ***SAT 2.0*** und ist auch nicht in Planung.

2.4 Bedienung und Funktionalität

Dieser Abschnitt versucht, die Bedienung von ***SAT 2.0*** zu erläutern und geht dabei auf die Details der jeweiligen Komponente ein. Es werden sämtliche Aspekte der Sicherheitsanalyse geschildert und die Ursachen möglicher Fehlermeldungen erklärt.

Das Erscheinungsbild des Benutzerinterfaces der ***SAT2*** Komponenten wurde bewusst auf die englische Sprache ausgelegt, um einen möglichst breiten Anwenderbereich abzudecken. Da diese Arbeit jedoch auf Deutsch formuliert ist, sei darauf hingewiesen, dass in dieser Beschreibung die Begriffe deutsch gehalten sind und die englischen Originalbezeichnungen nur bei Bedarf in Klammer mit angeführt werden.

2.4.1 Der SAT Controller

Der ***SAT*** Controller ist die Schaltzentrale des Systems (siehe Abbildung 2.4.1.1). Da sich diese Applikation der Microsoft Management Console als Framework bedient, muss sie als Snap-In registriert werden (siehe Kapitel 2.3). Um die Anwendung zu starten, muss zuerst eine Instanz der MMC aufgerufen werden. Dies wird beispielsweise erreicht, indem man im Windows Startmenü unter dem Punkt „Ausführen" das Kommando „mmc" eingibt.

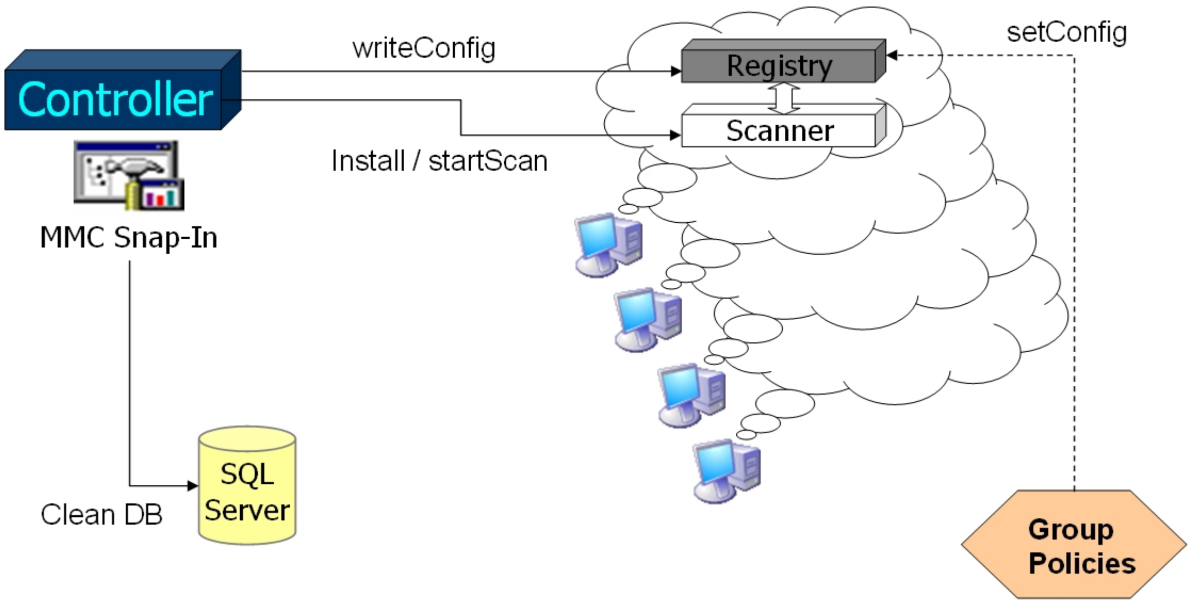


Abbildung 2.4.1.1: Schema des SAT2 Controllers

In der Management Console wird ein Snap-In über das Menü „Datei" - „Snap-In hinzufügen/entfernen" geladen. Die ***SAT*** Snap-Ins werden im Fenster „Eigenständig" mittels der „Hinzufügen…" Schaltfläche im Konsolenstamm der MMC integriert. Das Ausführen dieser Schaltfläche bewirkt zunächst die Aktivierung eines Dialogfensters, das sämtliche registrierten Snap-Ins, die der MMC zur Verfügung stehen, auflistet. Der als „SAT2Controller" angeführte Eintrag muss ausgewählt und durch Bestätigung - mittels der „Hinzufügen" Schaltfläche des Dialogfensters - aktiviert werden.

Das Laden des ***SAT*** Controllers bewirkt das Ausführen eines Abfragedialogs (siehe Abbildung 2.4.1.2), dessen Zweck es ist, dem Benutzer die Möglichkeit zu geben, den Host Computer für den Controller zu wählen. Diese Funktion ermöglicht es dem Administrator, auf die Controllerdaten eines fremden Rechners, sofern dieser im Netz verfügbar ist, zuzugreifen. Da diese Daten in der Registry des jeweiligen Rechners, der als Controller gewählt wurde, gespeichert werden, kann die Administration des ***SAT2*** Systems von einem beliebigen Knoten im Netzwerk erfolgen und wird trotzdem zentral - am Controller Rechner - gespeichert.

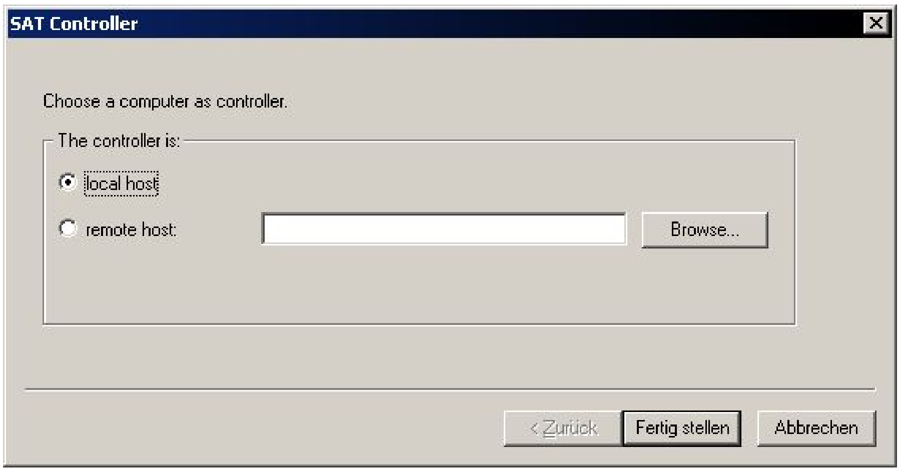


Abbildung 2.4.1.2: Host Computer Auswahldialog

Der Controller präsentiert sich zu Beginn als einzelnes Symbol unter dem Konsolenstamm, sofern er das erste Mal auf dem entsprechenden Host initialisiert wurde. Bei erneutem Aufruf des Snap-Ins werden die Controllerdaten aus der Registry des Controller Hosts ausgelesen und der Zustand des Systems, der mit dem Beenden der letzten Sitzung bestand, wieder hergestellt. Der Controller Wurzelknoten stellt die Ausgangsbasis des Snap-Ins dar und bietet zwei Funktionen an, die über das Kontextmenü gestartet werden können.

Die erste Funktion, die im Kontextmenü des Wurzelknotens aufscheint, ist „clear scan version“. Das Ausführen dieser Funktion bewirkt das Erscheinen eines Dialoges, der als Datenbankinterface dient. Um dieses Interface nutzen zu können, muss hier in der dafür vorgesehenen Eingabezeile der Name des Rechners mit dem ***SAT*** Datenbankserver angegeben werden. Besteht eine gültige Verbindung zu diesem Server, so werden die Versionen der Scandaten, die sich in der ***SAT*** Datenbasis befinden, in einem Pull-down-Feld aufgelistet. Die Schaltfläche „Clear this scan version“ bewirkt, dass die Scandaten der ausgewählten Version gelöscht werden. Diese Funktion ist bewusst ins ***SAT*** System integriert worden, um sicherzustellen, dass nur Scanversionen aus der Datenbank gelöscht werden, die der Benutzer für nicht mehr relevant hält. Somit obliegt die Entscheidung über die zu behaltenden Scanversionen und folglich über die Größe der Datenbank dem Administrator.

Als zweite Funktion im Kontextmenü bildet der „add Scan“ Befehl, der im Unterpunkt „Neu“ zu finden ist, die Verwaltungsbasis des Controllers. Dieser Befehl bewirkt wiederum das

Erscheinen eines Dialogs. In diesem Dialog (siehe Abbildung 2.4.1.3) kann ein neuer Scan Organisationsknoten erzeugt und konfiguriert werden. Ein Organisationsknoten kann als Container für einzelne Computer im Netz verstanden werden und ermöglicht das Konfigurieren und in weiterer Form das Starten des ***SAT*** Scanners auf den darunter liegenden Rechnern. Dieser Knoten wird eine Ebene unter dem Wurzelknoten eingehängt und dient zur übersichtlichen Verwaltung der geplanten Scanroutinen.

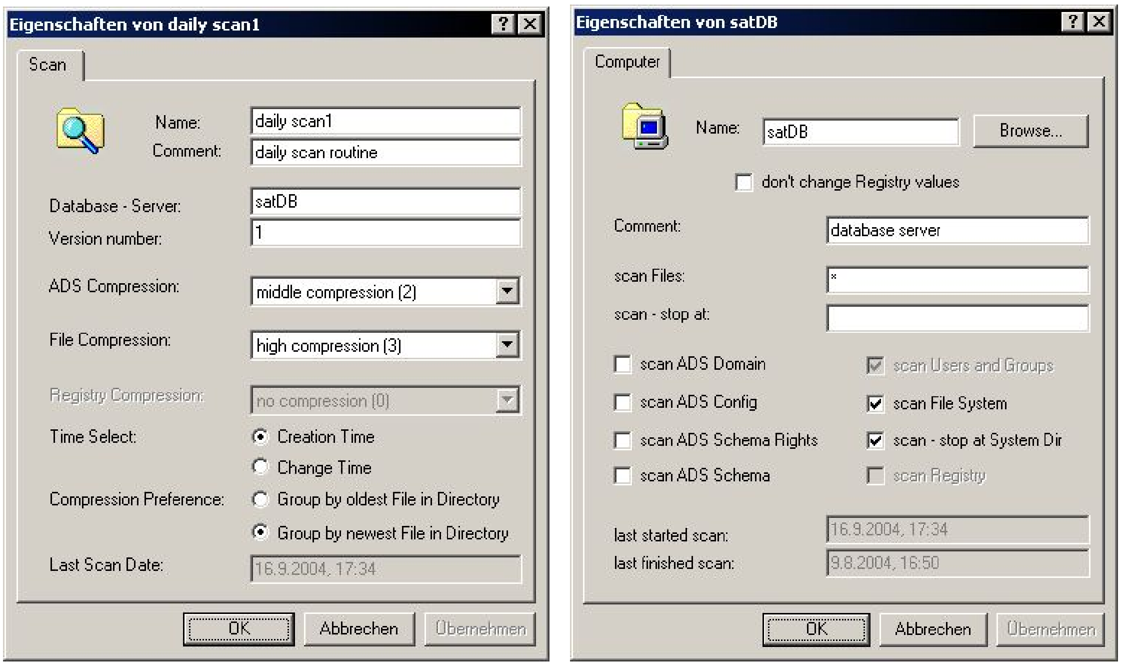


Abbildung 2.4.1.3 und 2.4.1.4: Konfigurationsdialoge für Scanknoten und Computerknoten

Der Scan Organisationsknoten verfügt wiederum über eine Reihe von Funktionen. Um einen Computerknoten im Scanknoten einzufügen, bietet das Kontextmenü im Unterpunkt „Neu" den Befehl „add computer" an, der den Konfigurationsdialog für den Computerknoten (siehe Abbildung 2.4.1.4) aufruft.

Mit dem Befehl „start Scan” werden eine Reihe von Aktionen ausgeführt. Zunächst überprüft der Controller, ob er jeden einzelnen der Computer im Scanknoten erreichen kann und ob auf diesen der ***SAT2*** Dienst installiert ist. Ist dies nicht der Fall, so registriert der Controller den Dienst am entsprechenden Rechner. Da der Dienst unter dem Konto *SATAdmin* gestartet wird, muss bei der Erstregistrierung das Passwort des Kontos angegeben werden, das der Controller in diesem Fall abfragt. Ist der ***SAT2*** Dienst auf dem Rechner installiert, so werden die eingestellten Scanoptionen in die Registry des Zielrechners übertragen und der Dienst wird schließlich gestartet. Das Symbol des Computerknotens wird dem Status des Dienstes angepasst.

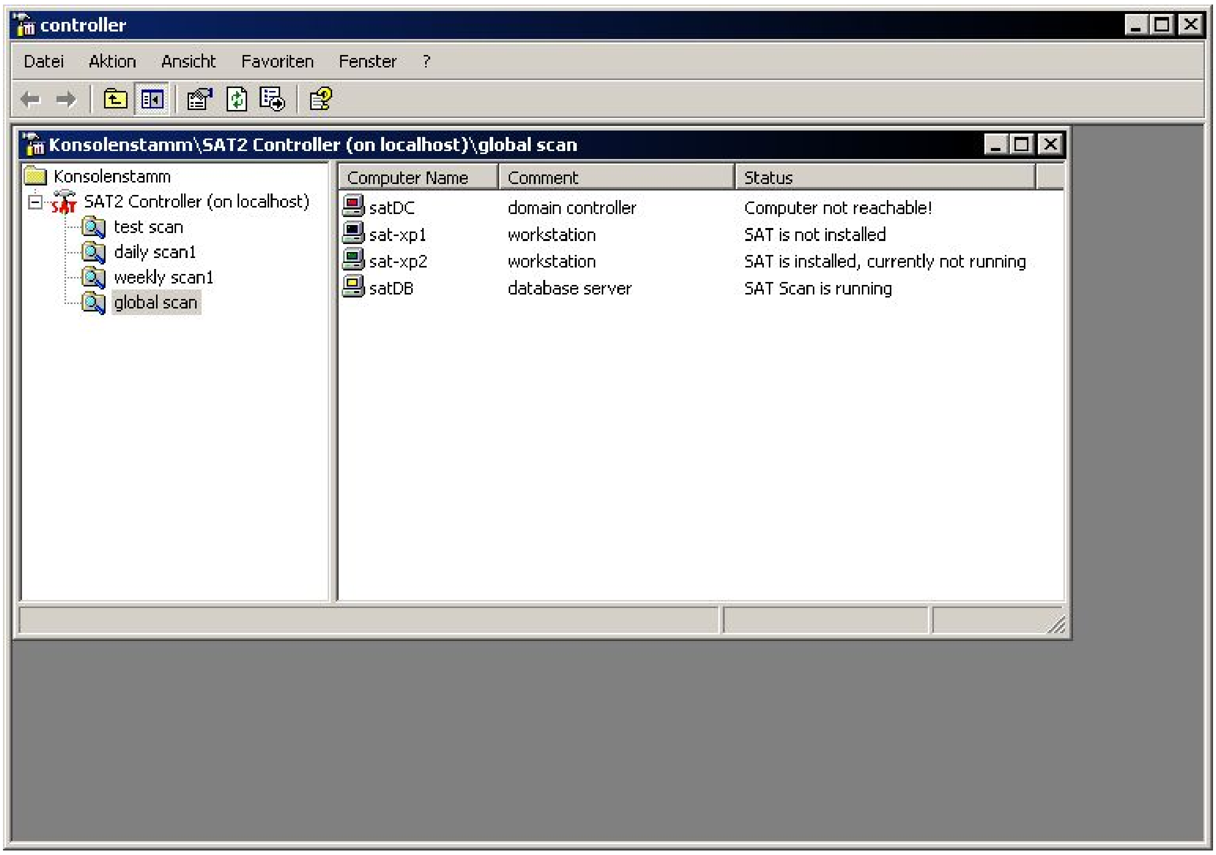


Abbildung 2.4.1.5: Controller mit Scan- und Computerknoten

Durch „remove Scan" wird der Organisationsknoten aus dem Snap-In entfernt und gelöscht. Die Funktion „check all Computers" initiiert eine Abfrage, die den Status des Scannerdienstes auf den einzelnen Rechnern im Organisationsknoten kontrolliert und die Symbole der Computerknoten (siehe Abbildung 2.4.1.5) entsprechend der im Folgenden aufgeführten Zustände einfärbt:

- *Rotes Computersymbol*:
  Der angegebene Computer kann nicht erreicht werden.

- *Schwarzes Computersymbol*:
  Der Computer konnte erreicht werden, jedoch ist der ***SAT2*** Dienst nicht installiert.

- *Grünes Computersymbol*:
  Der ***SAT2*** Dienst ist bereits auf dem Computer installiert, jedoch gerade nicht aktiv.

- *Gelbes Computersymbol*:
  Der ***SAT2*** Dienst läuft gerade auf dem Computer.

Die Konfigurationsoptionen, die zum einen im Scanknoten- und zum anderen im Computerknotendialog einstellbar sind, beziehen sich zum größten Teil auf den ***SAT*** Scanner und sollen im Folgenden kurz erläutert werden:

Einstellung	Registry Schlüssel	Funktionalität
Scanknoten:		
Name	*ScanName*	Name des Scan Organisationsknotens
Comment	*ScanKommentar*	Kommentar zum Scanknoten
Database - Server	*dbServer*	Name des Datenbankservers (NetBios oder DNS Name)
Version number	*Version*	Versionsnummer der Scanroutine
ADS Compression	*adsCompression*	Stufe der ADS Kompression
File Compression	*ntfsCompression*	Stufe der NTFS Kompression
Registry Compression (inaktiv)	*regCompression*	Stufe der Registry Kompression
Time Select	*creationTime*	NTFS- bzw. ADS-Scanner liest wahlweise den Zeitpunkt der Erstellung einer Datei oder den des letzten Zugriffs aus
Compression Preference	*comprPref*	Gruppierung in der NTFS Kompression (siehe Kapitel 2.4.3)
Last Scan Date	*lastScanDate*	Datum der zuletzt ausgeführten Scanroutine
Computerknoten:		
Name	*ComputerName*	Name des Computers (NetBios oder DNS Name)
don't change Registry values	*dontChange*	Anweisung an den Controller, dass die bestehenden Konfigurationswerte am Computer nicht geändert werden
Comment	*ComputerKommentar*	Kommentar zum Computerknoten
scan Files	*ntfsStartPoints*	zu scannende Laufwerke im NTFS (durch ; getrennt ohne Leerzeichen, oder * für alle Festplattenlaufwerke)
scan - stop at	*ntfsEndPoints*	Verzeichnisse, die vom Scan ausgenommen werden (durch ; getrennt ohne Leerzeichen)
scan ADS Domain	*scanAdsDomain*	aktiviert den ADS-Scanner, DomainNC-Container wird ausgelesen
scan ADS Config	*scanAdsConfig*	aktiviert den ADS-Scanner, Configuration-Container wird ausgelesen
scan ADS Schema Rights	*scanAdsSchemaRights*	aktiviert den ADS-Scanner, Schema-Container wird ausgelesen
scan ADS Schema	*scanAdsSchema*	aktiviert den ADS-Scanner, Schema wird analysiert
scan Users and Groups (inaktiv)	*scanUser*	aktiviert den CGU-Scanner; wird vorausgesetzt
scan File System	*scanNTFS*	aktiviert den NTFS-Scanner
scan - stop at System Dir	*scanStopAtSystemDir*	nimmt das System Verzeichnis aus dem NTFS Scan aus
scan Registry (inaktiv)	*scanRegistry*	aktiviert den Registry-Scanner; noch nicht implementiert

last started scan	*scanLastStarted*	Datum des letzten Scan Starts
last finished scan	*scanLastFinished*	Datum der Beendigung des letzten Scans

Tabelle 2.4.1.1.: Konfigurationsoptionen des SAT2 Scanners

Die Registry Schlüssel, wie sie in der Tabelle 2.4.1.1 aufgeführt sind, werden in der *Windows* Registry jedes Rechners, auf dem der ***SAT2*** Dienst gestartet wird, auf die in den Dialogen konfigurierten Werte gesetzt. Der Schlüsselzweig, unter dem diese Einträge zu finden sind, lautet:

- **HKEY_LOCAL_MACHINE\Software\Fim\Sat2**

Als besonderes Administrationsfeature wurde im Controller eine Funktion angedacht, die in der Konfigurationsoption „don't change Registry values" realisiert wurde. Ist diese Option aktiv, so werden die eingestellten Werte vorm Starten des Scanners nicht in die Registry des jeweiligen Computers übertragen, sondern sie werden in dem Zustand belassen, in dem sie „vor Ort" sind. Dies ermöglicht dem Administrator den Einsatz von Group Policies. Mittels dieser Group Policies wird das Konfigurieren von Sicherheitseinschränkungen wahlweise über ein eigenes Snap-In oder über ein einziges Skript, ein so genanntes Administrative Template, realisiert. Nähere Details zu diesem Thema sind unter [W2kA00/S.393 ff] bzw. unter [Ise00/S.261] nachzulesen.

2.4.2 Der SAT Viewer

Der ***SAT*** Viewer ist die Visualisierungskomponente des ***SAT2*** Systems. Die Implementierung dieser Komponente stellt den größten Teil des praktischen Aspekts dieser Diplomarbeit dar und soll deshalb im Folgenden detailliert beschrieben werden. Die theoretischen Grundlagen, die für die Umsetzung der angewandten Sicherheitsanalysen relevant sind, werden in Kapitel 3 und 4 ausführlicher behandelt.

Definiertes Ziel der ***SAT*** Viewer Komponente ist es, seinem Anwender eine benutzerzentrierte Sicht auf die Berechtigungen, die den Zugriff auf die im Netzwerk befindlichen Objekte regeln, zu bieten. Dazu werden vom Viewer die Daten der Sicherheitsanalyse der Scannerkomponenten (siehe Kapitel 2.4.3) herangezogen. Da es sich beim derzeitigen ***SAT 2.0*** System um ein Analysewerkzeug handelt, werden die Sicherheitsschwachstellen lediglich aufgezeigt, können aber nicht direkt korrigiert werden. Zu diesem Zweck muss der Administrator nach wie vor gebräuchliche Tools wie zum Beispiel den Windows Dateiexplorer heranziehen.

An dieser Stelle sei noch erwähnt, dass das ***SAT2*** System in erster Linie als Anwendung für Netzwerkadministratoren gedacht ist, die mit der Materie der Rechtevergabe in *Windows* Netzwerken vertraut sind. In diesem Sinne ist die *Viewer* Komponente auf die Visualisierung der wesentlichen Aspekte der Berechtigungsanalyse fokussiert und setzt Kenntnisse der Netzwerkadministration voraus.

Da der ***SAT*** Viewer auch als MMC Snap-In Applikation realisiert wurde, muss er ebenfalls in einer Instanz der Management Console gestartet werden. Auf das Laden eines Snap-Ins wird an dieser Stelle nicht mehr näher eingegangen, sondern auf das Kapitel 2.4.1 verwiesen. Angemerkt sei hier noch, dass im Dialog „Snap-In Hinzufügen“ der Eintrag „SAT2Viewer“ gewählt werden muss.

Wird der ***SAT*** Viewer geladen, so wird versucht, eine Verbindung zur Datenbank aufzubauen. Dabei wird der in der lokalen Registry konfigurierte Wert (siehe Kapitel 2.4.1) verwendet. Ist dieser nicht vorhanden, so wird eine Fehlermeldung ausgegeben. Eine Fehlermeldung entsteht auch, wenn der Datenbankserver nicht erreicht werden kann.

Besteht eine gültige Verbindung zur Datenbank, so präsentiert sich der ***SAT2*** Viewer in der Management Console als ein Wurzelknoten, unter dem sich 3 Module befinden, die in Abbildung 2.4.2.1 abgebildet sind und in den folgenden Abschnitt näher erläutert werden.

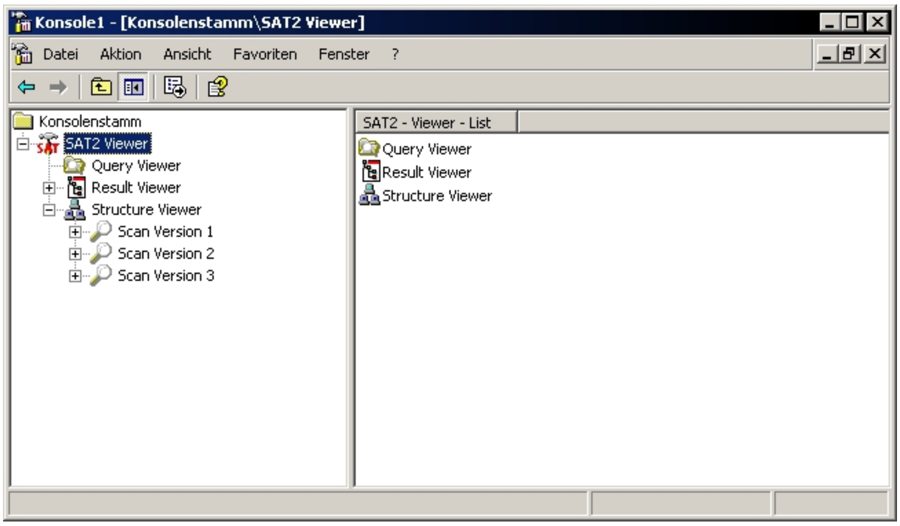


Abbildung 2.4.2.1: Überblick der Viewer Module

2.4.2.1 Der Query Viewer

Ausgangspunkt einer jeden ***SAT2*** Sicherheitsanalyse ist der Query Viewer. Dieses Modul dient dazu, Abfragen zu erstellen, zu verwalten und übersichtlich darzustellen. Auf diese Weise können wichtige Abfragen, die für die Sicherheit des Netzwerks essentiell sind, jederzeit erneut durchgeführt werden.

Um eine Abfrage zu erstellen, bietet das Kontextmenü des Query Viewers im Unterpunkt „Neu“ die Funktion „New Query…“ an. Das Ausführen dieser Funktion ruft einen Konfigurationsdialog (siehe Abbildung 2.4.2.1.1) auf, der zur Einstellung der gewünschten Analyseoptionen dient.

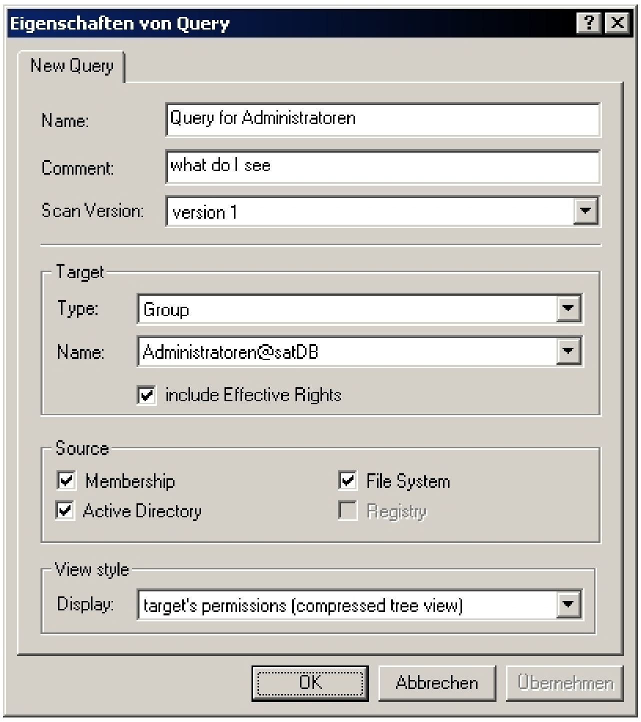


Abbildung 2.4.2.1.1: New Query Dialog

Zunächst kann die Abfrage mit einem Namen und einem Kommentar versehen werden, wobei diese Einträge lediglich zur Wahrung des persönlichen Überblicks des Anwenders gedacht sind. Als nächste Konfigurationsoption findet man ein Pull-down-Feld mit den in der Datenbank verfügbaren Versionen der durchgeführten Scans vor. Mittels dieser Option können beispielsweise Differenzanalysen durchgeführt werden, in denen die Rechte eines so genannten Security Principals (vgl. dazu Kapitel 3.1) über mehrere Scanroutinen hinweg verfolgt und ausgewertet werden. Zusätzlich existieren in diesem Konfigurationsfeld noch zwei Platzhaltereinträge - „earliest version“ und „latest version“ -, die ermöglichen sollen, dass eine Abfrage bewusst auf die erste bzw. letzte Version der Scandaten ausgerichtet abgelegt wird, unabhängig von der tatsächlichen Identifikationsnummer.

In der Target Sektion des Konfigurationsdialogs wird das Ziel der Analyse konkretisiert. Hier wird eine grundsätzliche Aufteilung der möglichen Security Principals in die drei Typen Computer, Gruppe (Group) und Benutzer (User) durchgeführt. Wird ein Typ ausgewählt, so erscheinen im darunter liegenden Pull-down-Feld die möglichen Security Principals zur Auswahl, die in der entsprechenden Scan Version diesem Typ zuzuordnen sind. Die Formatierung in diesem Name-Feld ist nach dem Schema „Security Principal@Computername“ bzw. im Falle des Computer Typs lediglich „Computername“ aufgebaut. Den Abschluss dieses Bereichs bildet das Kontrollkästchen „include Effective Rights“, das darüber entscheidet, ob in der Analyse des ausgewählten Security Principals auch die effektiven Berechtigungen, die sich aus bestehenden Gruppenzugehörigkeiten aufsummieren, mit einbezogen werden.

Der Source Abschnitt bietet die Auswahlmöglichkeit zwischen den Quelldaten, die von den einzelnen Scannerkomponenten ausgelesen wurden. Das „Membership“ Kontrollkästchen aktiviert die Mitgliedschaftsanalyse, die auf den Daten des CGU-Scanners operiert. Der „Active Directory“ Eintrag bietet folglich die Analyse der Daten des ADS-Scanners an. Durch das „File System“ Kontrollkästchen kann eine Analyse der NTFS-Scannerdaten initialisiert werden. Als vierten Punkt ist hier zusätzlich noch die Auswertung des Registry-Scanners vorgesehen. Da diese Scannerkomponente leider bisweilen noch nicht implementiert wurde, ist der „Registry“ Eintrag vorerst deaktiviert.

Abschließend entscheidet der View Style Abschnitt über den Modus der Analyse, wobei dieser nur Auswirkungen auf die NTFS- bzw. ADS-Analyse hat. In diesem Pull-down-Feld werden 7 verschiedene Modi angeboten, die in Abschnitt 2.4.2.2 noch ausführlicher erklärt werden.

Wurde die Abfrage nun sachgemäß erstellt, so erscheint ein Eintrag in der Result Pane Ansicht des Query Viewer (siehe Abbildung 2.4.2.1.2**)**. Die dort vorzufindenden Einträge dienen der Verwaltung der verschiedenen Analysen. Mittels des „Start Query“ Eintrags, den man über das Kontextmenü des Result Pane Knotens erreicht, kann die entsprechende Abfrage nun gestartet werden. Als Resultat dieser Aktion wird ein Subknoten mit den zuvor ausgewählten Analysen im Result Viewer generiert. Dies wird noch zusätzlich dadurch angezeigt, dass der Eintrag in der Statusspalte der Abfrage von „Not Executed!“ auf „Completed.“ gesetzt wird.

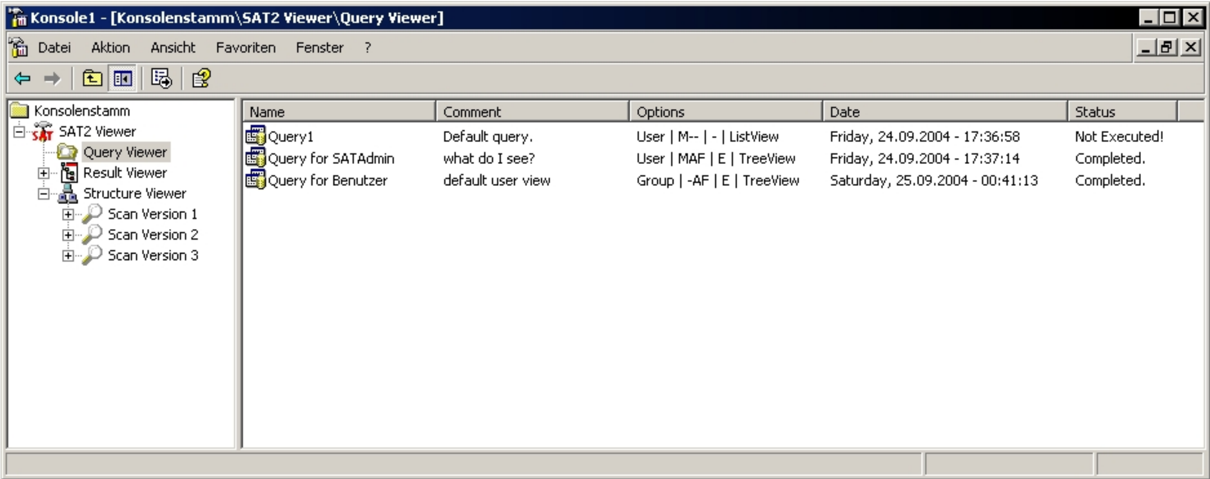


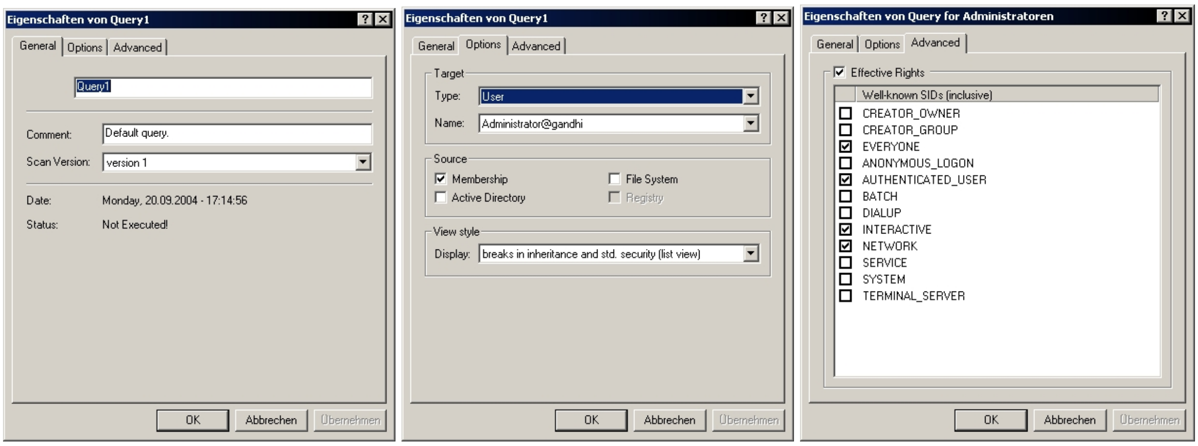
Abbildung 2.4.2.1.2: Query Viewer

Neben dem Namen, dem Kommentareintrag und der eben erwähnten Statusspalte gibt es noch weitere Abfragedetails, die zur übersichtlichen Darstellung herangezogen werden. In der Spalte „Date“ wird das Erstellungsdatum der Abfrage angezeigt. Die Spalte „Options“ liefert Aussage darüber, welche Analyseoptionen in der Abfrage formuliert wurden, und versucht diese in komprimierter Form nach dem in Tabelle 2.4.2.1.1 erläuterten Schema darzustellen. Dabei wurde besonderes Augenmerk darauf gelegt, dem Benutzer die Unterschiede der erstellten Abfragen möglichst einfach und übersichtlich vor Augen zu führen.

Sparte	Erklärung
Ziel:	**C** … Computer
	G … Group
	U … User
Quelle:	**M** … Membership
	A … Active Directory
	F … File System
Rechte:	**E** … include Effective Rights
Darstellungsart:	**L** … List View
	T … Tree View

Tabelle 2.4.2.1.1: Komprimierte Darstellung der Analyseoptionen

Um eine bereits erstellte Abfrage zu manipulieren bzw. deren Einstellung nochmals zu kontrollieren, besteht die Möglichkeit, einen Eigenschaftsdialog (siehe Abbildung 2.4.2.1.3, 2.4.2.1.4 und 2.4.2.1.5) aufzurufen. Dies geschieht wahlweise mittels Doppelklick auf den Abfrageeintrag, der bearbeitet werden soll oder mittels Aufruf des „Eigenschaften“ Eintrags im Kontextmenü. Eine Option, die sich aus Übersichtsgründen nur in diesem Dialog anbietet, ist die Konkretisierung des Eintrags „Effective Rights" unter dem Karteireiter „Advanced" (siehe Abbildung 2.4.2.1.5). Hier kann zusätzlich zur Aktivierung bzw. Deaktivierung der Einbeziehung der effektiven Benutzerrechte genau angegeben werden, welche „Wellknown SIDs" (vgl. Kapitel 3.1.1) in die Gruppenanalyse mit einfließen. Dabei werden nicht alle dem Betriebssystem bekannten SIDs angeführt, sondern nur jene, die im Kontext der Sicherheits-analyse sinnvoll erscheinen.



Abbildungen 2.4.2.1.3, 2.4.2.1.4 und 2.4.2.1.5: Eigenschaftsdialog einer Abfrage

2.4.2.2 Der Result Viewer

Wurde eine Abfrage im Query Viewer entsprechend konfiguriert und gestartet, so erscheint unter dem Result Viewer Knoten ein Subknoten, der als Namen die Bezeichnung der Abfrage führt. Dieser Queryknoten wird mit zwei weiteren Einträgen - dem Ziel der Abfrage und dem Datum der Ausführung - dargestellt, um auch hier eine gewisse Überschaubarkeit zu wahren.

Unter einem Queryknoten können drei Analyseknoten aufscheinen, die entsprechend der Konfiguration der Abfrage generiert werden:

- Membership Analysis
- Active Directory Analysis
- NT File System Analysis

Diese drei Knoten werden in ihrer Result Pane Repräsentation nochmals mit dem Ziel der Analyse, das diesmal in Zielname (target) und Zielort (location) aufgeteilt ist, und dem dazugehörigen SID angeführt.

Der Membership Analysis Knoten repräsentiert die Auswertung der Gruppenmitgliedschaften des angegebenen Ziels. Zum einen werden hier Gruppen aufgezählt, in denen der abgefragte Security Principal Mitglied ist. Zum anderen kann man hier weitere Security Principals verzeichnet sehen, die ihrerseits wiederum Mitglied im zu analysierenden Security Principal, sofern es sich dabei um eine Gruppe handelt, sind. Neben der Information, ob es sich bei einem Eintrag um ein Mitglied oder eine übergeordnete Gruppe handelt, werden weiters Name, Kommentar und DNS-Name (falls vorhanden) des Security Principals, auf den sich der Eintrag bezieht, angeführt.

In der Spalte „Type“ wird darauf eingegangen, ob sich die Mitgliedschaft direkt oder indirekt errechnet. Eine direkte Mitgliedschaft stellt die „normale“ Gruppenzugehörigkeit beispielsweise eines Benutzer A zur Gruppe X dar. Es kann jedoch auch möglich sein, dass eine Gruppe X Mitglied einer Gruppe Y ist und somit wäre der Benutzer A, der direktes Mitglied von X ist, folglich indirektes Mitglied von Y.

Schließlich wird unter der Spalte „Member SID“ angeführt, welcher SID dem Eintrag respektive dem Security Principal zuzuordnen ist. Besondere Behandlung wird an dieser Stelle den weiter oben bereits erwähnten Wellknown SIDs zuteil. In manchen Fällen kann es vorkommen, dass es sich bei dem Eintrag um eine Gruppe handelt, die als Wellknown SID im Betriebssystem per Default existiert - man spricht hier auch von so genannten „Built-In Groups“. Beispiele dafür wären die Gruppen „Administratoren“ oder „Benutzer“. Ist dies der Fall, so wird der angezeigte „Member SID“ mit dem SID des Computers qualifiziert, auf dem diese Standardgruppe ausgelesen wurde. Dies führt dann zu einer Darstellung der Form „Computer SID\Standardgruppe SID“. Neben dem entwicklungstechnischen Vorteil dieser Repräsentation dient die Qualifizierung zusätzlich noch einem praktischen Aspekt, da dadurch mehr Informationen gegeben werden und die Analyse nachvollziehbarer wird. Zusätzlich sei noch erwähnt, dass die Wellknown SIDs, die über den Eigenschaftsdialog im Query Viewer einbezogen werden können - beispielsweise die Gruppe „Jeder“ („Everyone“) -, in dieser Analyse nicht explizit angeführt werden, da diese im CGU-Scanner nicht direkt ausgelesen werden.

Der Active Directory Analysis Knoten und der NTFS Knoten unterliegen dem gleichen Aufbauschema. Der einzige oberflächliche Unterschied zwischen diesen beiden Analysetypen besteht darin, dass der NTFS Knoten als direkte Subknoten noch vor dem Ordnerknoten die

Laufwerksknoten der ausgelesen NTFS Wurzelverzeichnisse einhängt. In nomenklatorischer Hinsicht ergeben sich zwischen beiden Typen wiederum leichte Differenzen. Um die Knoten zu bezeichnen bedient man sich in der NTFS Analyse der Termini Ordner und Dateien, in der ADS Analyse wird von Containern und Objekten gesprochen. Da dem Autor eine allgemeine Formulierung wichtig scheint, werden in weiterer Folge die ADS Termini vordergründig Verwendung finden. Weitere semantische Unterschiede bestehen in den Berechtigungen, die auf den Objekten bzw. Containern gelten. Auf diese wird im Kapitel 3.2 näher eingegangen. Aufgrund der aufgeführten Ähnlichkeiten der ADS und NTFS Knoten sind die weiteren Ausführungen auf die Dateisystemanalyse abstrahiert, aber dennoch auf beide Typen zu beziehen.

Allgemein betrachtet präsentiert die Dateisystemanalyse der ***SAT2*** Viewer Komponente eine Übersicht über alle Objekte und Container, die auf den gescannten Computern existieren, und versucht diese für den Anwender sinnvoll aufzubereiten. Dabei wird eine Ansammlung von relevanten Daten für jedes Objekt bzw. für jeden Container in den Spalten des Result Pane aufgeführt. Die erste Spalte ist der obligatorische Namenseintrag, der den jeweiligen Knoten identifiziert. Weiters wird jeder Knoten mit der Spalte „date“ versehen, die je nach der ***SAT2*** Controller Konfiguration „Time Select“ (siehe Kapitel 2.4.1) das Erstellungs- bzw. das letzte Modifikationsdatum wiedergibt. Der Bereich „Object Type“ repräsentiert den Dateisystemtypen des Knotens, der im Falle der NTFS Analyse entweder mit „FSDirectory“ oder mit „FSFile“ abgebildet wird. Bei der ADS Analyse können beliebig viele Dateisystemtypen existieren, weshalb auch in der Datenbank eine Tabelle mit dem Namen „objTypes“ zu finden ist (vgl. Kapitel 3.3), die einzig dem Zweck dient, diese Typen zu erfassen und die korrekte Bezeichnung im Viewer anzuführen, um die Semantik der ADS Knoten zu verdeutlichen. Die nächste Spalte „Breaks Inh./StdSec.“ trifft darüber Aussage, ob der Knoten entweder die Vererbung unterbricht oder vom Standard Security Schema (siehe Kapitel 3.1.5) abweicht. Unter dem Standard Security Schema sind hier die Berechtigungen gemeint, die das Dateisystem per Default auf die neu angelegten Objekte überträgt.

Der Kern der benutzerzentrierten Sicherheitsanalyse steckt jedoch in den Spalten „Permissions“ und „Inherit Permissions“. „Permissions“ beschreibt die Berechtigungen, die dem analysierten Security Principal auf dem entsprechenden Knoten zugewiesen wurden. Die Darstellung dieser Berechtigungen ist auf eine komprimierte Form ausgelegt, die der Überschaubarkeit dient und im Kapitel 3.2 genauere Erläuterung findet. Analog dazu zeigt die

Spalte „Inherit Permissions“ Berechtigungen an, die vom aktuellen Knoten - sofern es sich dabei um einen Container handelt - auf dessen Subknoten vererbt werden, wobei hier zwischen den vererbten Rechten auf Containern („C:“) und Objekten („O:“) unterschieden wird. Wahlweise kann noch eine zusätzliche Berechtigungsspalte in die Ansicht eingefügt werden, die unter dem Namen „Read-/Write-Permission“ aufscheint und als eine reduzierte Darstellung der ursprünglichen „Permissions“ Spalte eine überschaubarere Auswertung bieten soll.

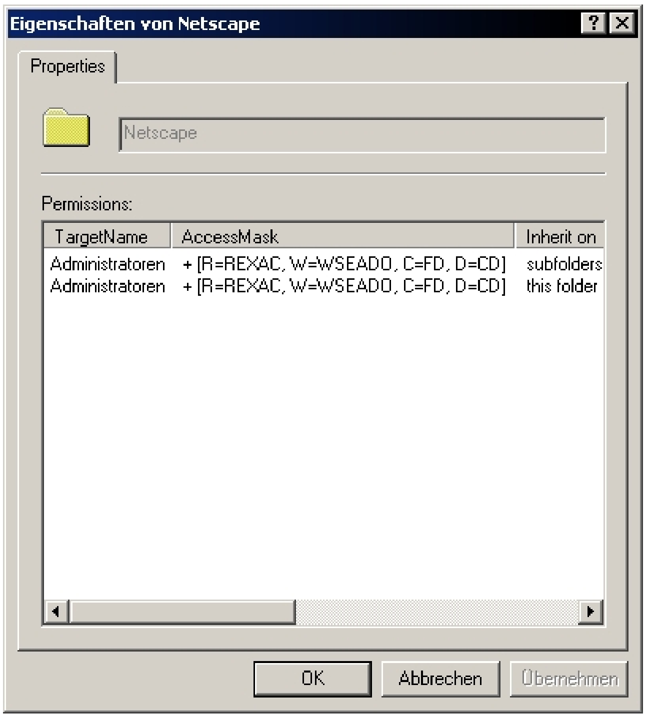


Abbildung 2.4.2.2.1: Eigenschaftsdialog eines Objekts

Um die exakte Ausprägung der Berechtigungen im Viewer anzuzeigen, bedient man sich des Eigenschaftsdialogs (siehe Abbildung 2.4.2.2.1), der über das Kontextmenü des zu untersuchenden Knotens aufrufbar ist. In diesem Dialog werden sämtliche ACE Einträge (siehe Kapitel 3.1) aufgelistet, die Bezug auf den analysierten Security Principal haben, der unter „TargetName“ aufgeführt ist. Neben dem „AccessMask“ Eintrag, der in Abhängigkeit davon, ob es sich bei dem ACE Eintrag um ein Verbot (Deny) oder eine Erlaubnis (Allow) handelt, mit einem Minus oder einem Plus als Präfix dargestellt wird, scheint weiters noch die Spalte „Inherit on“ auf. In dieser Spalte werden die Vererbungsstrategien angezeigt, auf die der ACE Eintrag zu beziehen ist, wie man es auch aus gebräuchlichen *Windows* Applikationen kennt. Die Spalte „FromStdSec“ weist darauf hin, ob der ACE Eintrag als Teil des Standard Security Schemas ausgelesen wurde. Schließlich wird in „ObjectType“ in manchen Spezialfällen der

ADS Analyse auf einen Objekttyp verwiesen, auf den sich der ACE Eintrag bezieht. Die „InheritedObjectType“ Spalte, die ebenfalls nur in der ADS Analyse relevant ist, kann einen Objekttyp beinhalten, der den ACE Eintrag erben kann.

Die Darstellung der einzelnen Knoten in der NTFS bzw. ADS Analyse hängt in erster Linie von dem in der Abfrage konfigurierten Analysemodus ab. Der Result Viewer bietet in Summe 7 verschiedene Analysemodi an:

- breaks in inheritance and std. security (list view)
- breaks in inheritance and std. security (compressed list view)
- target´s permissions (tree view)
- target's permissions (compressed tree view)
- target´s permissions (extended tree view)
- target´s permissions (compressed extended tree view)
- target´s permissions excl. defaults (extended tree view)

Generell kann hier zwischen 3 verschiedenen Darstellungsweisen unterschieden werden. Die erste Darstellungsweise, auf die durch den Zusatz „list view" hingewiesen wird, beschreibt eine „flache Liste", die keinen Verzeichnisbaum aufbaut, sondern alle Objekte als Endknoten in die Liste einhängt. Diese Art der Analyse fokussiert sich auf das Aufzeigen von Ausnahmen im Dateisystem. Im Konkreten bedeutet dies eine Auflistung all jener Objekte, die entweder die Vererbung unterbrechen und / oder vom Standard Security Schema abweichen. Da in einer Sicherheitsanalyse vor allem Abweichungen vom Regelfall Relevanz haben, ist diese Listenansicht darauf ausgelegt, dem Administrator mögliche Ansatzpunkte für Sicherheitslöcher aufzuzeigen.

Da bei der „list view“ Ansicht die eigentliche Dateisystemstruktur abstrahiert wird, ist diese Darstellungsweise um zwei zusätzliche Spalten erweitert worden. In der Spalte „full path“ findet man den genauen Dateisystempfad inklusive des Knotennamens. Die Spalte „source“ enthält den NetBios- bzw. den DNS-Namen des Hosts, falls ein solcher verfügbar ist. Diese beiden Spalten dienen vordergründlich der Nachvollziehbarkeit der Dateisystemanalyse in der „list view“ Ansicht.

Zur optischen Aufbesserung der Dringlichkeit von gefundenen Schwachstellen wird im ***SAT2*** Viewer eine „Ampelstrategie“ verfolgt. Dies wird durch die farbliche Darstellung der Objektsymbole realisiert, die getreu der Semantik der Ampelfarben abgestuft sind. Dabei ist die Interpretation der Einfärbung immer auf den jeweiligen Analysemodus zu beziehen. Im Falle der „list view“ Modi werden die Symbole nach den Berechtigungen, die direkt ad Objekt bzw. ad Container gelten, eingefärbt. Ein graues Symbol bedeutet, dass der analysierte Security Principal keine Zugriffsrechte auf das entsprechende Objekt besitzt. In direktem Gegensatz dazu verdeutlicht ein rotes Symbol, dass der Security Principal über sämtliche Berechtigungen verfügt und somit volle Kontrolle über das Objekt innehat. Ein grünes Symbol soll dem Anwender signalisieren, dass lediglich Leserechte vorliegen. Gelb bedeutet schließlich, dass Schreib- bzw. Änderungsrechte existieren. An dieser Stelle sei erneut auf das Kapitel 3.2 verwiesen, in dem die im Viewer verwendeten, komprimierten Rechte in die tatsächlichen, vom Betriebssystem vergebenen Rechte aufgeschlüsselt werden.

Die zweite Darstellungsweise wird als „tree view" Ansicht bezeichnet. Wie der Name schon sagt, baut diese Analyseart einen Verzeichnisbaum auf. Dieser ist dem regulären Dateisystemaufbau nachempfunden und wird nach dem Top-Down Analyseprinzip generiert. Im Falle der Benutzerberechtigungsanalyse soll dieses Prinzip so verstanden werden, dass besagter Verzeichnisbaum „on demand" aufgebaut wird. Das heißt, die Subknoten eines Containers werden erst im Result Pane eingefügt und analysiert, sobald der Container ausgewählt bzw. expandiert wird. Diese Ansicht lässt das Inspizieren der NTFS bzw. ADS Verzeichnisstrukturen auf den gescannten Computern zu und verdeutlicht die Berechtigungen, die der abgefragte Security Principal auf die Objekte und Container besitzt.

Die Symboleinfärbung passiert analog zur „list view“ Darstellung. Auch hier werden die Symbole entsprechend den Berechtigungen gesetzt, die ad Objekt respektive ad Container für den Security Principal gelten. Dieser Modus der Darstellung eines Dateisystems kann somit als eine Art erweiterter Dateiexplorer verstanden werden, der mit Bezug auf einen Security Principal dessen Berechtigungen bzw. dessen Sichtweise verdeutlicht und symbolisiert.

Klarer Vorteil der Top-Down Analyse ist die Geschwindigkeit der Auswertung, da nur vergleichsweise wenig Objekte in die Analyse mit einzubeziehen sind. Da diese Ansicht jedoch den Nachteil aufweist, dass der ***SAT2*** Anwender mitunter zig Ebenen im Verzeichnisbaum durchsucht, um schließlich festzustellen, dass keine Sicherheitsbedenken vorliegen, sollte sie wohl eher lediglich zu Übersichtszwecken herangezogen werden.

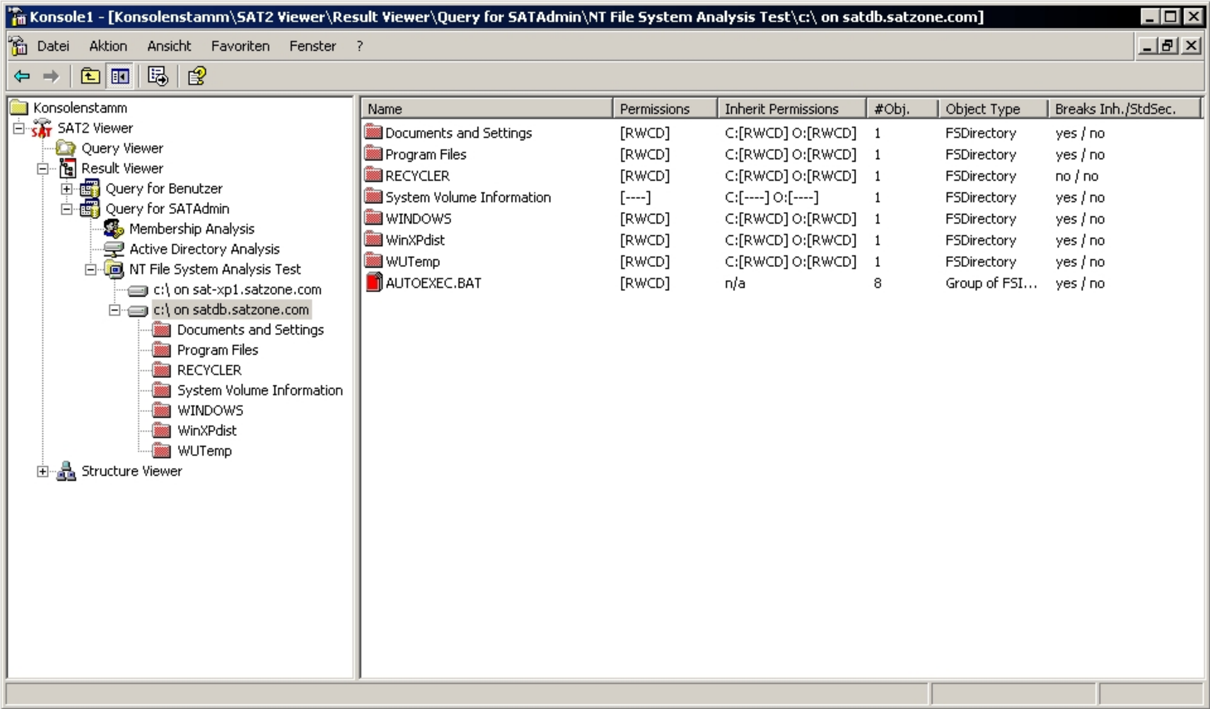
Um bei der Suche nach Sicherheitsmängeln Zeit und Nerven zu sparen, wurde die dritte Darstellungsweise konzipiert, die unter dem Begriff „extended tree view" zur Auswahl angeboten wird. Im Aufbau ähnlich der „tree view" Ansicht ist diese Analyseart durch das Bottom-Up Prinzip geprägt. Die Bottom-Up Analyse versucht, gesamte Verzeichniszweige bis in die unterste Dateisystemebene in die Auswertung mit einzubeziehen und die Berechtigungen, die für den zu analysierenden Security Principal gesetzt sind, aufzusummieren. Somit besteht die Möglichkeit, jeden Knoten des Verzeichnisbaumes in Abhängigkeit der Sicherheitsmerkmale seiner Söhne (siehe dazu das Baumkonzept in Kapitel 4.4) darzustellen. Dies bedeutet für die „extended tree view" Ansicht, dass dem Administrator permanent vor Augen geführt wird, über welche Berechtigungen ein Security Principal „irgendwo" unterhalb eines Knotens maximal noch verfügt.

Die durch die Bottom-Up Analyse errechneten, kumulierten Berechtigungen resultieren schließlich in der Symbolfarbe, die semantisch wiederum der sicherheitstechnischen Relevanz entspricht. Der Unterschied zu den anderen 2 Modi besteht darin, dass die Symbole nun zusätzlich auf sämtliche Subknoten, falls vorhanden, Bezug nehmen. Rote Symbole bedeuten in diesem Sinne, dass der analysierte Security Principal Vollzugriff auf den aktuellen Knoten und / oder auf einen bzw. auf mehrere Subknoten innehat. Graue Symbole weisen darauf hin, dass keine Berechtigungen auf den zu untersuchenden Knoten inklusive sämtlicher eventuell vorhandener Subknoten gegeben sind. In der Bottom-Up Analyse bedeutet dies, dass sämtlichen Subknoten keine analytische Bedeutung mehr beigemessen werden muss. Somit sind in diesem Modus graue Symbole mit Endknoten gleichzusetzen und werden auch als solche dargestellt, unabhängig davon, ob es sich um Objekte oder Container handelt. Grüne Symbole weisen auf das Vorhandensein eines Leserechts auf dem aktuellen Knoten oder auf dessen Söhne hin. Besitzt der abgefragte Security Principal ein Schreibrecht darauf, so wird ein gelbes Symbol verwendet.

Da die Bottom-Up Analyse diffiziler und gründlicher arbeitet als die Top-Down Analyse, ist es nicht verwunderlich, dass hier Performanzprobleme auftreten können. Dies hängt in erster Linie von der Anzahl der Dateisystemobjekte ab, die in die Analyse mit einzubeziehen sind. In der Entwicklung des ***SAT2*** Viewer Prototypen hat sich diese Art der Analyse als Flaschenhals herauskristallisiert. Logischer Schritt war, die Überarbeitung und Optimierung dieses Analysemodus in Bezug auf die Effizienz der Datenbankabfragen und des grafischen Aufbaus in der Microsoft Management Console. Dieses Vorgehen brachte signifikante Verbesserung in die Bottom-Up Analyse, die den für Administratoren verständlicherweise interessantesten Zweig der benutzerzentrierten Sicherheitsanalyse darstellt, da sie punktgenau an die Sicherheitsschwachstellen heranführt. Obwohl etwas langsamer, ist sie der Top-Down Analyse gegenüber doch klar im Vorteil.

Wie aus der Auflistung der verschiedenen Analysemodi weiter oben ersichtlich wird, existiert für jede der drei Darstellungsmöglichkeiten ebenfalls eine „compressed" Variante. Hinter dieser Variante verbirgt sich ein Verfahren, das mehrere Endknoten, auf denen die gleichen Sicherheitsmerkmale für den abgefragten Security Principal gelten, in einem Knoten gruppiert und den „Object Type" Eintrag auf „Group of Items" korrigiert. Dies geschieht unabhängig von der möglichen Kompression, die für den Scanner konfiguriert werden kann, und fällt mit einer zusätzlichen Spalte „#obj" im Result Pane auf, die für die Anzahl der Elemente in der neu generierten Gruppe steht. Auf die Scanner Kompression wird gesondert im Kapitel 2.4.3 eingegangen. Das definierte Ziel dieser Viewer Kompression ist es, eine noch kompaktere Darstellung der Analyse zu erzielen, um selbst bei einer großen Anzahl von Knoten die Übersichtlichkeit zu wahren.

Eine Spezialform der komprimierten Bottom-Up Analyse ist der letzte Eintrag in der Auswahlliste der Analysemodi, der als „target´s permissions excl. defaults" verzeichnet ist (siehe Abbildung 2.4.2.2.2). Zusätzlich zur „normalen" Bottom-Up Analyse und der inkludierten Viewer Kompression werden in diesem Modus sämtliche Einträge unterhalb eines Knotens, die im Vergleich zu diesem keine Änderung in den Sicherheitsmerkmalen mehr darstellen, aus dem Verzeichnisbaum abstrahiert. Im Detail bedeutet dies, dass die Analyse überprüft, ob unter dem aktuellen Knoten Subknoten existieren, die von einem Default abweichen, der durch die auf den Knoten vererbten Berechtigungen gegeben ist. Werden keine Abweichungen gefunden, so kann davon ausgegangen werden, dass diese Subknoten keine sicherheitstechnische Relevanz mehr aufweisen. Dies hat ein Löschen der entsprechenden Subknoten aus dem darzustellenden Baum zur Folge. Durch diese Vorgehensweise wird das Ergebnis der Bottom-Up Analyse auf das Wesentlichste reduziert. Somit stellt dieser Modus die effizienteste Variante der 7 Analysemodi dar und bietet die übersichtlichste Darstellung in der benutzerzentrierten Sicherheitsanalyse.



Abbildungen 2.4.2.2.2: NTFS Analyse im Modus „target´s permissions excl. defaults“

Abschließend sei noch erwähnt, dass das ***SAT*** System den Besitzer eines Objekts nicht in die Rechteanalyse mit einbezieht. Da der Besitz eines Objekts (vgl. Kapitel 3.1.2) den „Owner“ zu Änderungen der ACL berechtigt, kann daraus ein Sicherheitsproblem resultieren, das in den Analysen des ***SAT*** bisweilen nicht berücksichtigt wird.

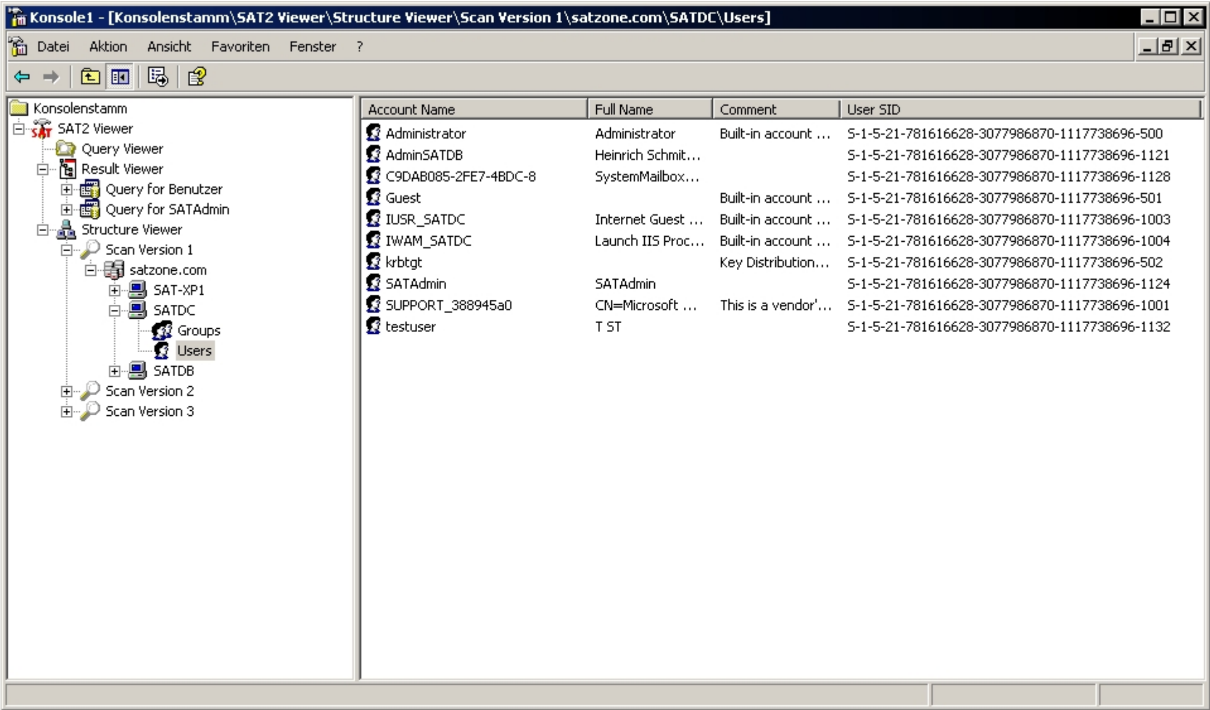
Ferner sei angemerkt, dass ein Security Principal, der aufgrund der Kompatibilität zu *Windows NT4* Rechte in einer Active Directory Domäne innehat, nur dann korrekt und vollständig analysiert werden kann, wenn ein ADS-Scan durchgeführt wurde. Ist dies nicht der Fall, so werden diese „*Windows NT4* SIDs“ weder in die NTFS noch in die ADS Analyse mit einbezogen.

2.4.2.3 Der Structure Viewer

Als drittes Modul der Viewer Komponente wurde der Structure Viewer ursprünglich dazu entwickelt, dem Anwender einen kurzen Überblick über die Struktur des gescannten Netzwerks vor Augen zu führen. Zu diesem Zweck wurde ein Konzept implementiert, das auf vier verschiedenen Knotentypen basiert (siehe Abbildung 2.4.2.3.1):

- Domänenknoten
- Computerknoten

- Gruppenknoten
- Benutzerknoten



Abbildungen 2.4.2.3.1: Knotenübersicht im Structure Viewer

Der Domänenknoten repräsentiert den Wurzelknoten der gescannten ADS-Domäne, falls eine solche vorhanden ist. Darunter liegen die Computerknoten, die den Rechnern in dieser Domäne entsprechen. Da es in gebräuchlichen *Windows* Netzen durchaus vorkommen kann, dass keine Domäne existiert bzw. dass es Rechner gibt, die der vorhandenen Domäne nicht angehören, ist es möglich, Computerknoten auf derselben Hierarchieebene vorzufinden, auf der die Domänenknoten eingehängt werden. Solche „stand-alone“ Computer sind abgesehen von ihrer Position äquivalent zu den Domänencomputern.

Die im Result Pane aufgeführten Informationen zu diesen beiden Knotentypen beschränken sich auf Identifikationsdaten. Die Spalte „Name“ zeigt im Falle eines Domänenknotens den vollen Namen der Domäne an. Handelt es sich um einen Computerknoten, so wird hier der NetBios Name angeführt. Die weiteren Spalten werden nur mehr von den Computerknoten verwendet. Unter „DNS HostName“ wird der vollständige DNS-Name des Computers angezeigt, sofern es einen entsprechenden DNS-Eintrag dafür gibt. Im Weiteren sind noch die Spalten „Local SID“ und „Domain SID“ zu finden, die SID Informationen zum Computer-knoten enthalten, da dieser auch als Security Principal fungieren kann. Der „Local SID“ steht für die Identifikationsnummer, die dem Computer bei der Installation des Betriebssystems zugewiesen wurde. Schließt sich der Computer einer Domäne an, so wird ein „Domain SID" vergeben, der den Computer innerhalb der Domäne eindeutig identifiziert und den SID Präfix der Domäne enthält.

Unter dem Computerknoten befinden sich die Container für die Gruppenknoten (Groups) und die Benutzerknoten (Users). In diesen Containern sind die lokalen Gruppen und Benutzer vorzufinden, es sei denn, der übergeordnete Computerknoten ist ein Domain Controller. In diesem Fall gehören die Gruppen und Benutzer der Domäne an.

Die zu diesen beiden Knotentypen angezeigten Details behandeln wiederum deren Identifikationsdaten. Die Spalte „Name", die sich bei den Benutzerknoten in „Account Name" und „Full Name" aufteilt, beschreibt wiederum die Kennung des jeweiligen Security Principals. „Comment" deutet auf den Kommentareintrag hin, den der Administrator beim Anlegen des Security Principals angegeben hat. Schließlich ist unter der Spalte „Group SID" bzw. „User SID" die Sicherheitsidentifikationsnummer aufgeführt, über die jeder Security Principal eindeutig bestimmt werden kann.

Im Laufe der Entwicklung des ***SAT2*** Viewers entstand eine zusätzliche Abstufung der Structure Daten. Da sich Netzwerkstrukturen auf kurze oder lange Sicht durchaus ändern können und da der ***SAT2*** Scanner über Versionierung verfügt, wurde der Knotenaufbau um einen Versionsknoten erweitert, der nunmehr den ursprünglichen 4 Knotentypen übergeordnet ist. Dies bringt den Vorteil mit sich, dass der Structure Viewer einen Überblick über die Migration des analysierten Netzwerks widerspiegelt. Somit wird nachvollziehbar, welche Computer in die Scanroutinen zu welchem Zeitpunkt aufgenommen wurden und ob der lokale SID des jeweiligen Rechners durch eine Neuinstallation des Betriebssystems geändert wurde. Auf diese Weise kann eine Rückverfolgung von SID Informationen, die das Betriebssystem nicht mehr auflösen kann, stattfinden.

Als zusätzliches Feature verfügt der Structure Viewer weiters über die Möglichkeit, eine neue Abfrage direkt über das Kontextmenü des gewünschten Security Principals zu initialisieren, da sich das Konfigurieren einer Abfrage bei zunehmender Zahl von Security Principals immer schwieriger gestaltet und man leicht den Überblick verlieren kann. Dieses Feature birgt den Vorteil, dass man so direkt den gewünschten Security Principal in der gewünschten Scanversion auswählen kann und nicht der Verwirrung von unzähligen, verschiedenen Versionen unterliegt. Somit stellt der Structure Viewer eine effektivere Basis für die Konfiguration einer neuen Abfrage da.

2.4.3 Der SAT Scanner

Da die Implementierung der Scannerkomponente nicht im Aufgabenbereich des Autors dieser Arbeit lag, sich aber im Laufe der Fertigstellung des ***SAT2*** etwaige Verbesserungen aufgedrängt haben, die vom Autor durchgeführt wurden, wird diese Komponente hier kurz und mit Bezug auf die Neuerungen abgehandelt.

Der ***SAT2*** Scanner ist ein *Windows* Dienst (vgl. dazu [W2kA00/S. 668]), der als solcher im Betriebssystem registriert werden muss. Dies geschieht durch den ***SAT2*** Controller, wie in Kapitel 2.4.1 nachzulesen ist. Wie jeder Dienst, so ist auch der Scanner im MMC Snap-In *services.msc* zu finden, das die Liste der lokalen Dienste aufzeigt und deren Verhalten verwaltet.

Jeder *Windows* Dienst läuft unter einem bestimmten Konto. Dies kann unter anderem auch das Konto *Local System* sein. Aufgrund sicherheitsrelevanter Überlegungen wird der Scanner Dienst jedoch nicht unter diesem mächtigen Systemkonto gestartet, sondern greift auf das zu diesem Zweck angelegte Konto *SATAdmin* zurück. Somit kann der Einsatz des ***SAT*** Scanners auch in größeren, kritischeren Netzen vertreten werden, da dieser Dienst durch das Konto *SATAdmin* in der Prozessliste nachvollziehbar und in seinen Kompetenzen einschränkbar bleibt.

Damit der Scanvorgang problemlos abläuft, muss der *SATAdmin* über ausreichend Zugriffsrechte verfügen, um domänenweit die Sicherheitsinformationen, die von den einzelnen Modulen des Scanners gesammelt werden, auslesen zu können. Hat der Scanner auf ein Verzeichnis bzw. auf einen Container keinen Zugriff, so wird ein Platzhaltereintrag in die Datenbank geschrieben, der in der Viewer Komponente als „?DIRECTORY_NOT_ACCESSIBLE?" Eintrag erscheint und darauf hinweist, dass dieses Verzeichnis oder dieser Container mangels Zugriffsrecht nicht analysiert werden konnte.

Die Konfigurationen des Scanner Dienstes sind in Kapitel 2.4.1 konkret beschrieben und werden in diesem Kapitel nicht mehr explizit behandelt.

Wie schon in Kapitel 2.2 erwähnt, ist der Scanner in 4 separate Module aufgeteilt:

- Computer-, Gruppen- und Benutzerscanner (CGU-Scanner)
- Active Directory Service Scanner (ADS-Scanner)
- NTFS-Scanner
- {Registry-Scanner}

Der CGU-Scanner stellt die Basis für die Analysen der restlichen Module dar. Sowohl der ADS-Scanner, als auch der NTFS-Scanner benötigen zuerst die Security Principal Referenzen, die durch die CGU-Analyse aufgebaut werden, um korrekt zu funktionieren. Somit ist die Verwendung dieses Moduls auch im Controller unumgänglich.

Konkret bedeutet eine CGU-Analyse, dass die Security Principals auf dem jeweiligen Rechner, auf dem der CGU-Scan ausgeführt wird, der Reihe nach ausgelesen und in die Datenbank geschrieben werden. Dabei werden zuerst die Computerinformationen bearbeitet, gefolgt von den Sicherheitsdaten der lokalen Gruppen und schließlich werden sämtliche lokale Benutzerkonten abgearbeitet. Um die Gruppenmitgliedschaftsanalyse der Viewer Komponente überhaupt durchführen zu können, müssen letztlich noch die Gruppenzugehörigkeiten in der Datenbank erfasst werden. Diese Zuordnung passiert als letzte Instanz der CGU-Analyse. Genauere Informationen zum CGU-Scanner sind in [Zar02/S.76 ff] nachzulesen.

Der Active Directory Service Scanner stellt das diffizilste Modul des ***SAT2*** Scanners dar. Im Gegensatz zum NTFS-Scanner sollte dieses Modul lediglich auf Rechnern ausgeführt werden, die als Domain Controller fungieren. Grundsätzlich kann der ADS-Scanner wiederum in zwei 2 Programmmodule unterteilt werden:

- ADS-Object Scanner
- ADS-Schema Scanner

Der ADS-Schema Scanner ist dafür zuständig, die diversen Objekttypen und die dazugehörige Standard-Security auszulesen. Der ADS-Object Scanner hingegen muss sämtliche Objekte inklusive der Zugriffskontrolllisten für jedes Objekt speichern.

Die Trennung zwischen ADS Objekten und dem ADS Schema eröffnet dem Scanner eine flexiblere Arbeitsweise. Falls in einer Domäne mehrere Domain Controller existieren, so werden die Active Directory Objekte auf diese Domain Controller aufgeteilt, das Schema hingegen wird generell repliziert, d.h., jeder Domain Controller enthält die gleiche Kopie des Schemas. Um nun einen effizienten ADS Scan durchzuführen, ist es sinnvoll, das Schema nur auf einem Domain Controller auszulesen.

Da die Datenmenge der ADS Analyse selbst in kleineren Systemen schon beträchtlich ist, wurden im ADS-Scanner drei Kompressionsstufen implementiert. Unter Kompression sind in diesem Zusammenhang sowohl das Zusammenfassen gleicher Elemente als auch die Abstraktion von vernachlässigbaren Details zu verstehen. Die Kompression der einzelnen Stufen funktioniert wie folgt:

- Kompressionsstufe 1: Zusammenfassung gleicher Objekttypen
  In dieser Stufe werden Objekte zu einem Datenbankeintrag zusammengefasst, wenn Folgendes gilt:
    - Die Objekte befinden sich in einem Container derselben Hierarchiestufe.
    - Container werden nur zusammengefasst, wenn sie keine Objekte enthalten.
    - Die Objekte müssen die gleiche *Access Control List* (vgl. Kapitel 3.1.3) aufweisen.
    - Die Objekte müssen vom gleichen Objekttyp sein.
- Kompressionsstufe 2: nur Ausnahmen werden gespeichert
  Diese Kompressionsstufe ist stark verlustbehaftet und speichert Objekte nur, wenn auf sie eine von den folgenden Aussagen zutrifft:
    - Die Objekte brechen die Vererbung der Berechtigungen auf.

    - Die Objekte weichen vom Standard Security Schema ab.

- Kompressionsstufe 3: Kompressionsstufe 1 und 2 werden kombiniert
  Diese Stufe führt zuerst die Kompressionsstufe 1 durch, danach die Kompressionsstufe 2. Daraus resultiert die höchstmögliche Kompression im ADS-Scanner.

Um konkretere Informationen zum ADS-Scanner zu erhalten, sei der Leser auf [Hel00/S.72 ff] verwiesen.

Der NTFS-Scanner, dessen Funktionalität ein vereinfachtes Subset des ADS-Scanners darstellt, kann auf jedem Rechner im Netz durchgeführt werden. Um dies durchzuführen, werden im Controller die auszulesenden Festplattenlaufwerke eingetragen. Es ist darauf zu achten, dass diese mittels Strichpunkt („;“) getrennt und ohne Leerzeichen angegeben werden. Wahlweise kann man auch als Wildcard das Sternsymbol („*“) verwenden, das als Platzhalter für sämtliche Festplattenlaufwerke auf dem zu scannenden Rechner dient.

Aufgabe des NTFS-Scanners ist es, die auf den Laufwerken existierenden Dateien und Ordner in die Datenbank zu übertragen. Zusätzlich müssen die Berechtigungen, die auf den ausgelesenen Elementen gelten, analysiert und gespeichert werden. Im Gegensatz zum ADS gibt es im NTFS lediglich zwei Objekttypen, Ordner und Dateien. Die Aufschlüsselung der auf diesen Typen existierenden Rechten ist in Kapitel 3.2 nachzulesen.

Analog zum ADS-Scanner bietet auch der NTFS-Scanner Kompressionsstufen an:

- Kompressionsstufe 1: Zusammenfassen von gleichen Objekten
  Diese Stufe gruppiert Objekte gleicher Hierarchieebene mit gleicher *Access Control List* (ACL) zu einem Eintrag, die dem gleichen Dateityp angehören, d.h. die gleiche Datei-Extension besitzen. Die Anzahl der gruppierten Dateien wird gespeichert und scheint im Viewer in der Spalte „#Obj“ auf. Der Name des Eintrags kann über die Controller Einstellung „Compression Preference“ wahlweise auf den Namen der ältesten oder der jüngsten Datei in der neu generierten Gruppe gesetzt werden.

- Kompressionsstufe 2: Zusammenfassen von Objekten mit gleicher ACL
  In dieser Stufe werden sämtliche Dateien eines Verzeichnisses, die über die gleiche ACL verfügen, zu einem Eintrag zusammengefasst, der als „*.*“ im Viewer aufscheint.

Der Beitrag des Entwicklers des NTFS-Scanners behandelt dieses Thema leider nicht ausführlicher. Er kann unter [Ach02] nachgelesen werden.

Die Implementierung des Registry-Scanners wurde aus Mangel an Projektmitarbeitern einstweilen ausgesetzt, obgleich auch der Registry-Scanner lediglich ein Subset des ADS-Scanners darstellt und der Einsatz des Scanners in den restlichen Teilen des ***Security Analysis Tools 2.0*** bereits vorbereitet wurde.

2.5 Sicherheitsanalyse in der Praxis

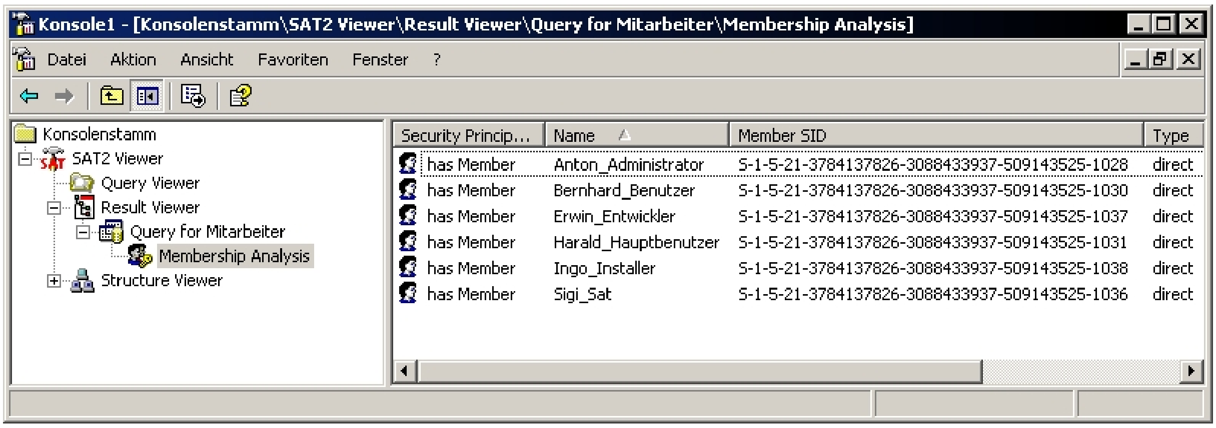
In diesem Abschnitt wird versucht, die in Kapitel 2.4.2 beschriebenen Analysen anhand von Beispielen aus der Netzwerkadministration näher zu erläutern, um die Funktionalität des ***SAT2*** Viewers zu demonstrieren. Zu diesem Zweck wurde ein Testnetzwerk konstruiert, das im Labor des Instituts für Informationsverarbeitung und Mikroprozessortechnik nachgebildet und analysiert wurde.

An dieser Stelle sei angemerkt, dass die Darstellung der Ergebnisse der Dateisystemanalyse NTFS optimiert ist. Die Ergebnisse der ADS Analysen werden analog zur NTFS Ansicht dargestellt, um ein einheitliches Benutzerinterface präsentieren zu können. In seltenen Fällen, wie etwa im Falle der Interpretation der so genannten „extended rights" (Näheres dazu in Kapitel 3.2.2), stößt diese Darstellungsweise auf ihre Grenzen und resultiert schließlich in der Abstraktion der ADS spezifischen Daten.

2.5.1 Das Testszenario

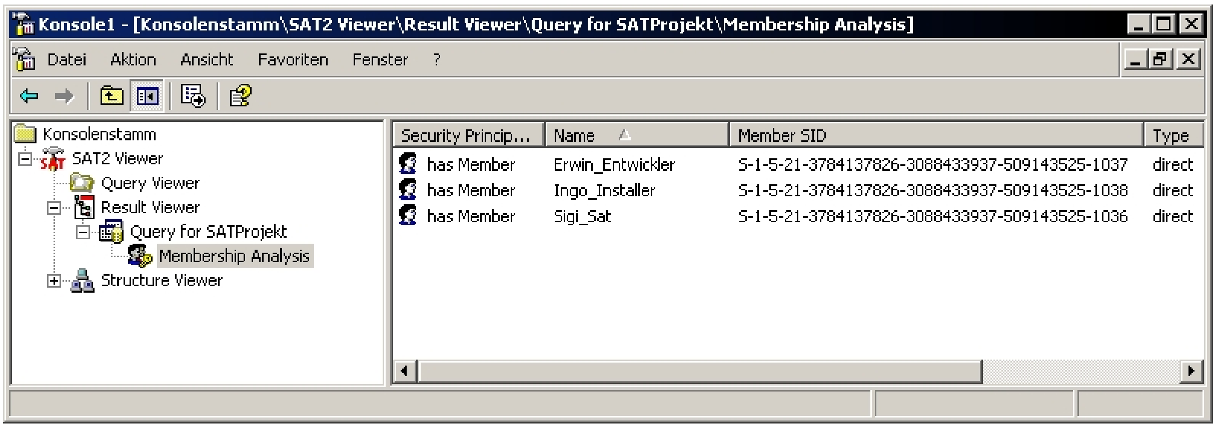
Um dem Leser ein realitätsnahes Szenario bieten zu können, das zum besseren Verständnis der Arbeitsweise des ***SAT2*** Systems führen soll, sind die Datei- und Benutzerstrukturen im Testaufbau einer fiktiven Softwarefirma nachempfunden, die neben anderen Projekten auch mit der Entwicklung eines Sicherheitsanalysesystems namens ***SAT2*** beschäftigt ist. Die

Angestellten dieser Firma werden durch die Benutzergruppe „Mitarbeiter“ repräsentiert (siehe die Mitgliedsanalyse in Abbildung 2.5.1.1) und werden in den weiteren Abschnitten dieses Kapitels als Akteure der Analysebeispiele auftreten.



Abbildungen 2.5.1.1: Mitgliedsanalyse der Gruppe Mitarbeiter

Als zweite analyserelevante Benutzergruppierung wird die Gruppe „SATProjekt“ verwendet (siehe Abbildung 2.5.1.2), die sich als Untermenge der Gruppe „Mitarbeiter“ versteht und diejenigen Benutzer beinhaltet, die im Projektszenario ***SAT2*** inbegriffen sind.



Abbildungen 2.5.1.2: Mitgliedsanalyse der Gruppe SATProjekt

Der Aufbau des Testnetzwerks, auf das sich die folgenden Analysen beziehen, wurde absichtlich so konzipiert, dass jeder verwendete Rechner eine dezidierte Funktion im ***SAT*** Projekt erfüllt. Zu diesem Zweck wurden drei Rechner installiert, die über einen 100 MBit Ethernet Switch miteinander verbunden wurden. Da die Größe des Netzwerks und die Anzahl der verwendeten Computer für die folgenden Auswertungen keinerlei Relevanz haben und das funktionale Modell des ***SAT2*** Systems mit 3 Rechnern adäquat abgedeckt werden kann, wurde auf eine Adaption der Testnetzes an die im Szenario vorgegebene Firmenstruktur verzichtet.

Als Active Directory Domain Controller und „Kernstück“ des Testnetzes wurde ein *Microsoft Windows 2003 Advanced Server* aufgesetzt und die Testdomäne „satzone.com“ eingerichtet. Zusätzlich zur ADS Funktionalität werden auf diesem Rechner, der die domänenweite Bezeichnung „satdc.satzone.com“ trägt, ein DHCP Server und ein DNS Server betrieben.

Der Datenbankserver *Microsoft SQL Server 2000* wurde auf dem zweiten Rechner mit dem DNS Namen „satdb.satzone.com“ installiert. Als Betriebssystem dient diesem Computer *Microsoft Windows XP Professional*, das mit dem Service Pack 1 auf den zum Zeitpunkt der Testphase aktuellsten Stand gebracht wurde.

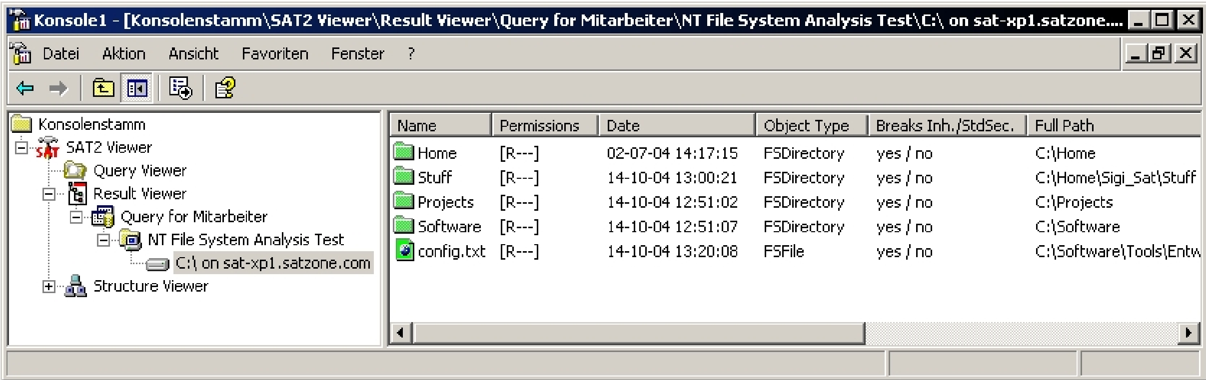
Der dritte Computer in der Domäne, der als „sat-xp1.satzone.com“ bezeichnet wurde, ist ebenfalls ein *Microsoft Windows XP Professional (SP1)* System. Dieser Rechner dient zum einen als File Server für das Firmennetzwerk und zum anderer als Host für den ***SAT2*** Viewer sowie den ***SAT2*** Controller. An dieser Stelle sei angemerkt, dass die Konstellation mit beiden ***SAT*** Snap-Ins auf einem Rechner für den erfolgreichen Betrieb des Systems nicht zwingend erforderlich ist, da Controller und Viewer als eigenständige Komponenten implementiert sind und somit auf beliebigen Rechnern im Netz verteilt sein können und funktionieren.

Die Controllerkonfiguration, die zum Auslesen der Analysedaten für die in den nächsten Abschnitten folgenden Beispiele Verwendung fand, wurde bereits in Kapitel 2.4.1 ausführlich beschrieben und wird in diesem Kapitel nicht mehr explizit erläutert.

2.5.2 Schwachstellenanalyse

Als Einführung in den praktischen Aspekt der benutzerzentrierten Sicherheitsanalyse wird zunächst der Modus „breaks in inheritance and std. security” veranschaulicht. Das Ziel dieses Darstellungsmodus ist, dem Administrator alle im System vorhandenen Abweichungen vom Regelfall vor Augen zu führen. Dies wird bewerkstelligt, indem eine Liste mit potenziellen Schwachstellen präsentiert wird, die im Zusammenhang mit dem analysierten Security Principal auftreten. Die Abschätzung der tatsächlichen Bedrohung, die von den einzelnen Einträgen in der Liste ausgeht, obliegt jedoch dem Administrator selbst.

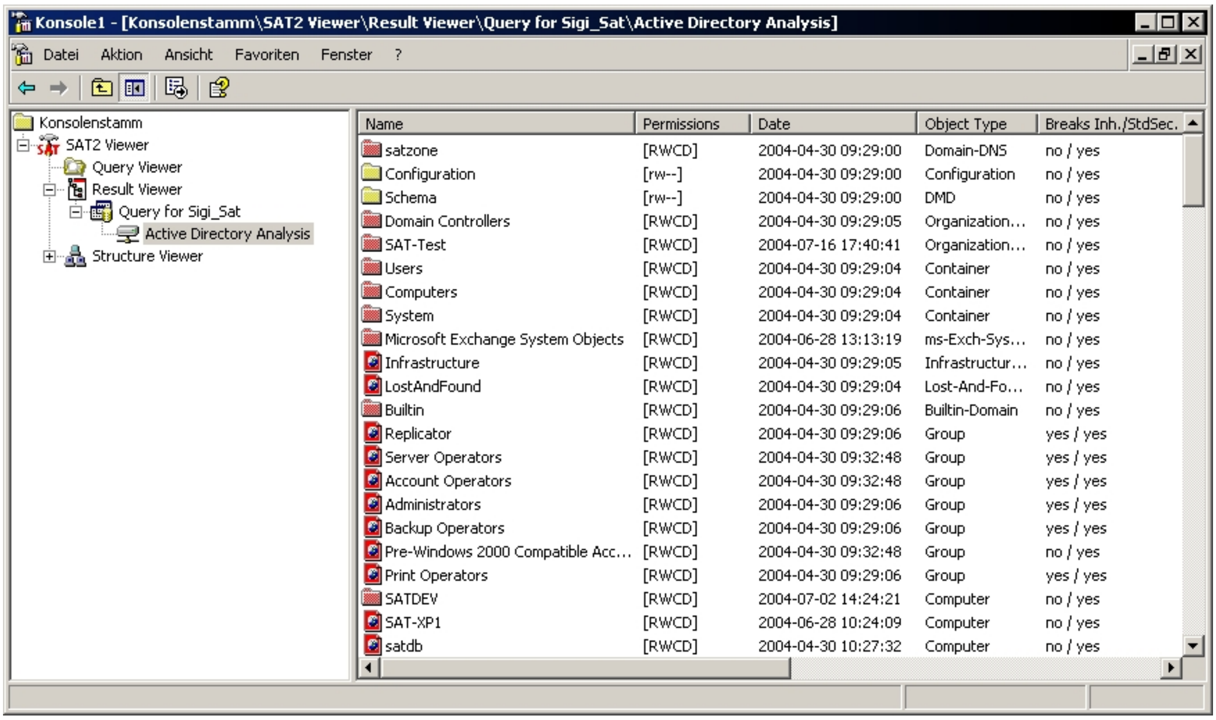
Als erstes Beispiel soll die Gruppe „Mitarbeiter“, der sämtliche Angestellte der fiktiven Softwarefirma angehören, einer NTFS Analyse unterzogen werden. Dazu wird eine neue Abfrage erzeugt, der die File System Daten als Quelle dienen und die nur auf die Gruppe „Mitarbeiter“ exklusive der effektiven Berechtigungen, die sich aus eventuellen Mitgliedschaften in anderen Gruppen herleiten, abzielt.



Abbildungen 2.5.2.1: NTFS Analyse im List View Modus

Das Ergebnis dieser Abfrage (siehe Abbildung 2.5.2.1) zeigt sämtliche Brüche in der Vererbung bzw. Abweichungen in der Standard Security auf, die durch die analysierte Gruppe im Dateisystem der gescannten Festplatten entstanden sind. Daraus geht hervor, dass besagte Gruppe lediglich auf einem Rechner im System, dem File Server „sat-xp1.satzone.com“, Leserechte hat. Da diese Einträge in keinem Widerspruch zu den firmeninternen Sicherheitsrichtlinien stehen, ist keinerlei administrativer Handlungsbedarf gegeben.

Das nächste Beispiel soll verdeutlichen, dass der Modus der Schwachstellenanalyse in vielen Fällen zu unübersichtlichen, und somit zu schwer interpretierbaren Ergebnissen führen kann. Basierend auf den Daten des Active Directory Scanners wird eine Abfrage erstellt, deren Ergebnis die Sicherheitsbrüche des Benutzers „Sigi_Sat“ inklusive seiner effektiven Berechtigungen, die sich aus der Mitgliedschaft in der Gruppe „Administrators“ und „SATProjekt“ ergeben, aufzeigt. Wie aus dem Ergebnis der Analyse (siehe Abbildung 2.5.2.2) zu schließen ist, existiert im firmeninternen Active Directory eine Unzahl an theoretischen Sicherheitslöchern bezüglich des analysierten Benutzers. Da die Abfrage jedoch indirekt auf die Gruppe „Administrators“ bezogen war, kann daraus weiters gefolgert werden, dass eine Vielzahl dieser Brüche der Vererbung bzw. der Standard Security durchaus gewollt ist und wiederum nicht näher untersucht werden muss.

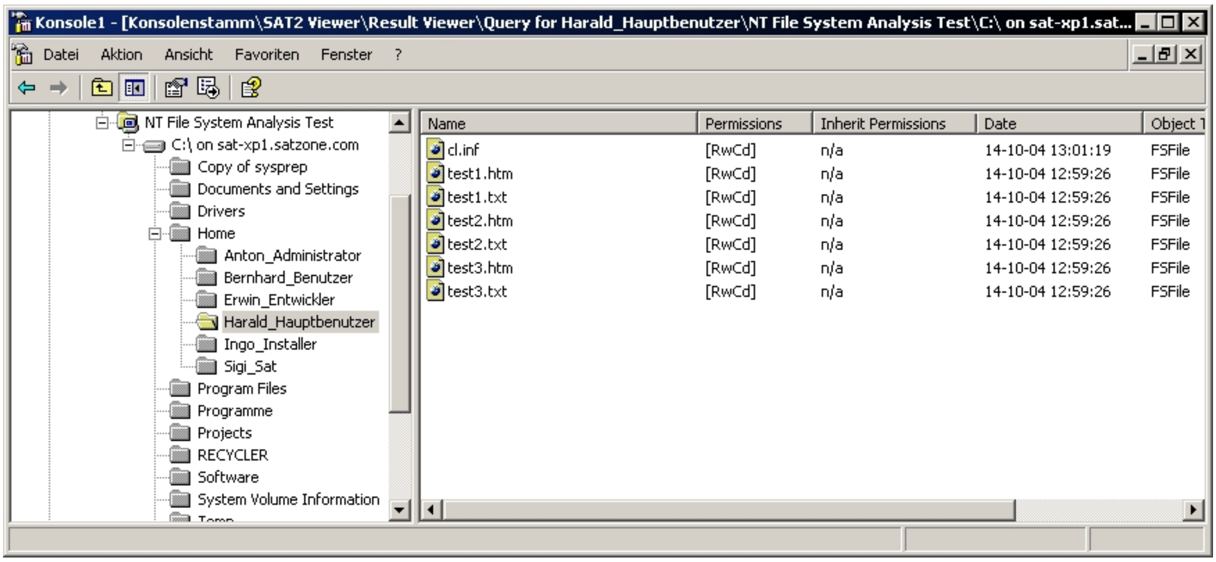


Abbildungen 2.5.2.2: ADS Analyse im List View Modus

2.5.3 Top-Down Analyse & Kompression

Dieser Abschnitt beschäftigt sich mit den verschiedenen Möglichkeiten der Datenkompression, die im ***SAT2*** System zur Verfügung stehen. Als Anzeigemodus wird die Tree View Darstellung „target´s permissions" verwendet, die im Folgenden sowohl in komprimierter als auch in unkomprimierter Form zum Einsatz kommt. Dieser Modus führt eine Top-Down Analyse der Scannerdaten durch und interpretiert die Berechtigungen ausschließlich direkt auf das jeweilige Objekt bezogen. Da diese Darstellungsweise einem erweiterten Dateiexplorer gleichkommt und die eigentliche Sicherheitsanalyse weitgehend dem Benutzer überlassen wird, sollte sie nur zu Übersichtszwecken verwendet werden, wie es in den folgenden Beispielen der Fall ist.

Im Mittelpunkt der folgenden Beispiele steht der Benutzer „Harald_Hauptbenutzer“. Es wird eine NTFS Analyse des Benutzerverzeichnisses von „Harald_Hauptbenutzer“ durchgeführt, das sich, wie allgemein üblich, auf dem File Server der Firma befindet. Um die Differenzen zwischen den verschiedenen Kompressionsmethoden aufzuzeigen, wird zunächst das Verzeichnis unkomprimiert analysiert. Das Resultat dieser Analyse ist in Abbildung 2.5.3.1 zu sehen.

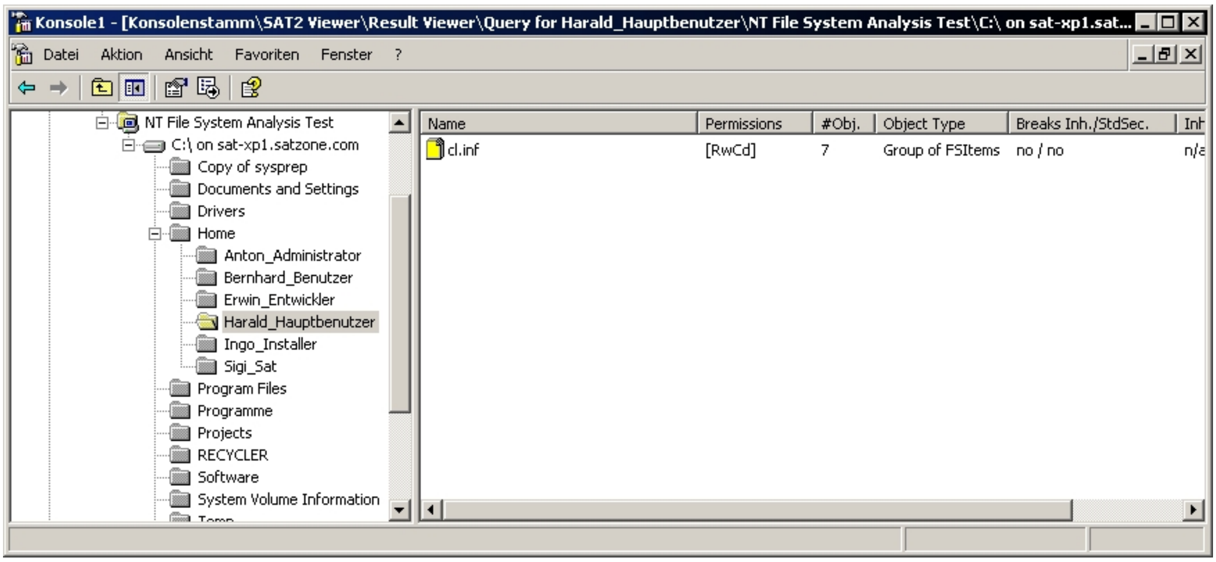


Abbildungen 2.5.3.1: Tree View unkomprimiert im Benutzerverzeichnis

In dem zu untersuchenden Verzeichnis befinden sich augenscheinlich 7 Dateien, die 3 verschiedene Dateitypen aufweisen. Wie weiters ersichtlich ist, besitzt der analysierte Benutzer auf jede Datei die gleichen Zugriffsrechte. Ausgehend von dieser Dateistruktur werden nun die Kompressionsmöglichkeiten beleuchtet.

Als erste Variante besteht die Möglichkeit, die Viewer Darstellungskompression anzuwenden. Dies muss zunächst in der jeweiligen Abfrage im Query Viewer durch Verwendung der „compressed" Variante des gewählten Darstellungsmodus eingestellt werden (siehe dazu Kapitel 2.4.2.1). Wird nun eine Analyse durchgeführt, so ergibt sich daraus das in Abbildung 2.5.3.2 dargestellte Bild.

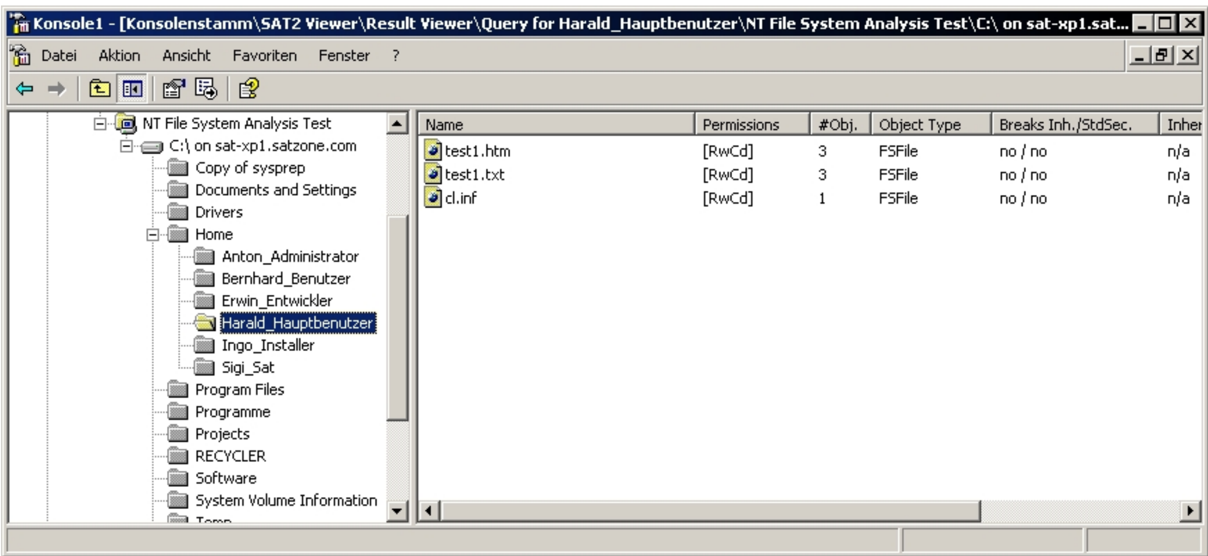
Da die Darstellungskompression des Viewers weitgehend einer Gruppierungsfunktion entspricht, die eine kompaktere und somit übersichtlichere Ansicht forciert, zeigt das Ergebnis lediglich einen Eintrag, der den Typ „Group of FSItems" repräsentiert und dessen „#Obj" Attribut die Gesamtanzahl der gruppierten Elemente widerspiegelt. Die grundlegende Regel, die hier Anwendung findet, besagt, dass sämtliche Elemente zusammenzufassen sind, die als Endknoten gelten und deren ACL Einträge in Bezug auf den untersuchten Security Principal gleich sind.



Abbildungen 2.5.3.2: Tree View komprimiert im Benutzerverzeichnis

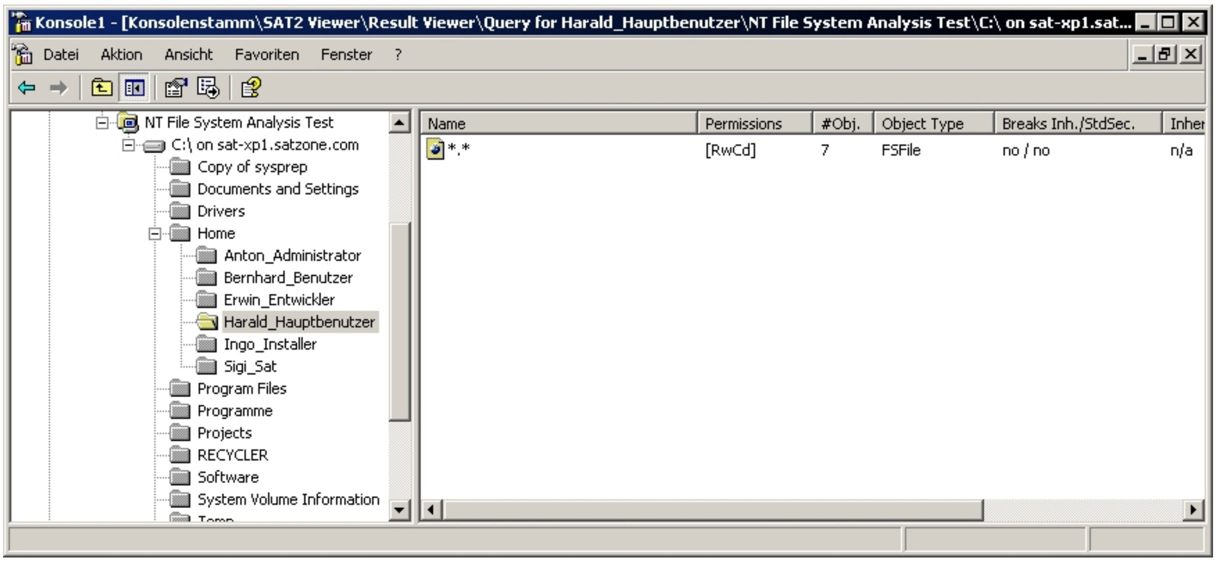
Eine andere Möglichkeit, um Datenkompression im ***SAT2*** System zu erreichen, bietet die Scanner Komponente an. Wie in Kapitel 2.4.3 genauer beschrieben wird, existieren sowohl für die NTFS als auch für die ADS Analyse verschiedene Kompressionsstufen.

Die erste NTFS Kompressionsstufe gruppiert Objekte mit identischer ACL zu einem Eintrag, sofern sie sich in der gleichen Hierarchieebene befinden und den gleichen Dateityp - in der NTFS Analyse bedeutet dies die gleiche Dateierweiterung - aufweisen. Eine auf solchen Daten durchgeführte Analyse resultiert in einer Übersicht, wie sie in Abbildung 2.5.3.3 dargestellt ist.



Abbildungen 2.5.3.3: Tree View, Scanner Kompression Level 1 im Benutzerverzeichnis

Die NTFS Kompressionsstufe 2 geht noch einen Schritt weiter. Weisen Endknoten in einem Verzeichnis identische ACLs auf, so werden diese unbeachtet ihres Dateityps durch einen einzelnen Eintrag ersetzt, der mit dem Platzhalter „*.*“ angeführt wird. In unserem konkreten Fallbeispiel ergibt sich für den Security Principal „Harald_Hauptbenutzer“ das in Abbildung 2.5.3.4 angeführte Analyseresultat.



Abbildungen 2.5.3.4: Tree View, Scanner Kompression Level 2 im Benutzerverzeichnis

Abschließend muss darauf hingewiesen werden, dass sich jede Variante der Kompression verständlicherweise mit erhöhter Programmlaufzeit bemerkbar macht. Wird die Darstellungskompression des Viewers angewandt, so verzögert sich der Aufbau eines Verzeichnisses proportional zur Anzahl der darin enthaltenen Elemente. Benutzt man Scannerkompression, so muss man mit einer längeren Laufzeit des Scannerdienstes rechnen.

Die Vorteile der Kompression haben jedoch auch ihre Gültigkeit. Einerseits ist die erhöhte Übersichtlichkeit der Analyse ein Zeit sparender Faktor im Umgang mit ***SAT2***. Zum anderen macht sich die Scannerkompression in der zu speichernden Datenmenge bezahlt, wie aus Tabelle 2.5.3.1 ersichtlich wird.

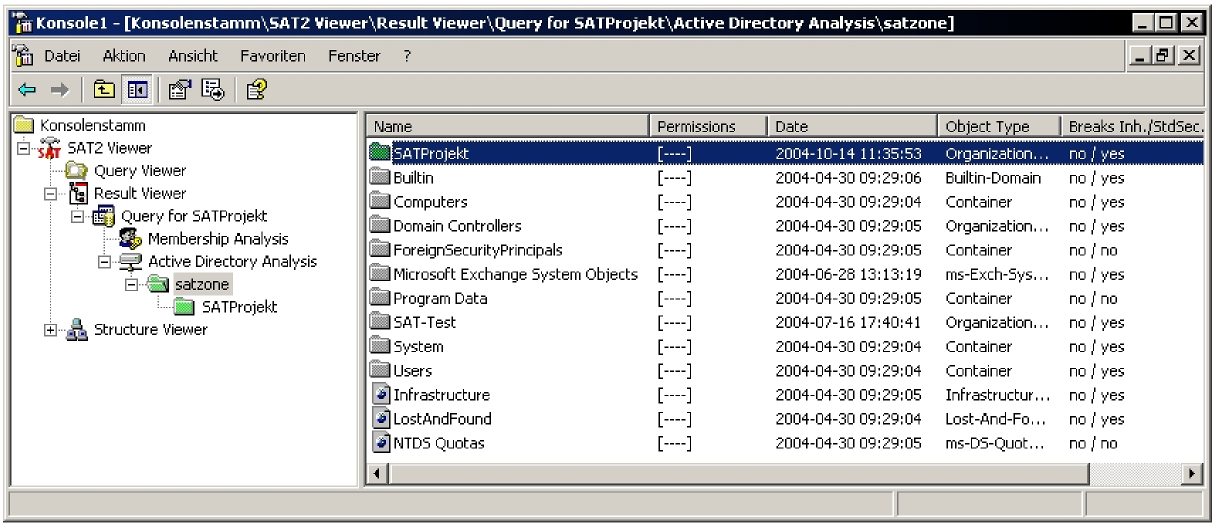
Modus	Anzahl NTFS Objekte
NTFS Level 0	**22556**
NTFS Level 1	**4131**
NTFS Level 2	**2670**

Tabelle 2.5.3.1: Anzahl der Datensätze in der SAT DB

2.5.4 Bottom-Up Analyse

Die Bottom-Up Analyse repräsentiert den „mächtigsten“ Modus des ***SAT2*** Viewers. Diese Analysevariante ist durch den Zusatz „extended tree view“ gekennzeichnet und versucht, dem Administrator in der Interpretation der dargestellten Berechtigungsbäume vorzugreifen indem errechnet wird, ob die Untersuchung eines Verzeichniszweigs noch weiteren Aufschluss bieten kann. Das bedeutet, dass die Extended Tree View Ansicht versucht, für die Sicherheitsanalyse irrelevante Teilzweige auszublenden. Daraus resultiert ein reduzierter Verzeichnisbaum, der durch die Einfärbung der Knoten verdeutlicht, wie dringend Untersuchungs- bzw. Handlungsbedarf am jeweiligen Knoten besteht.

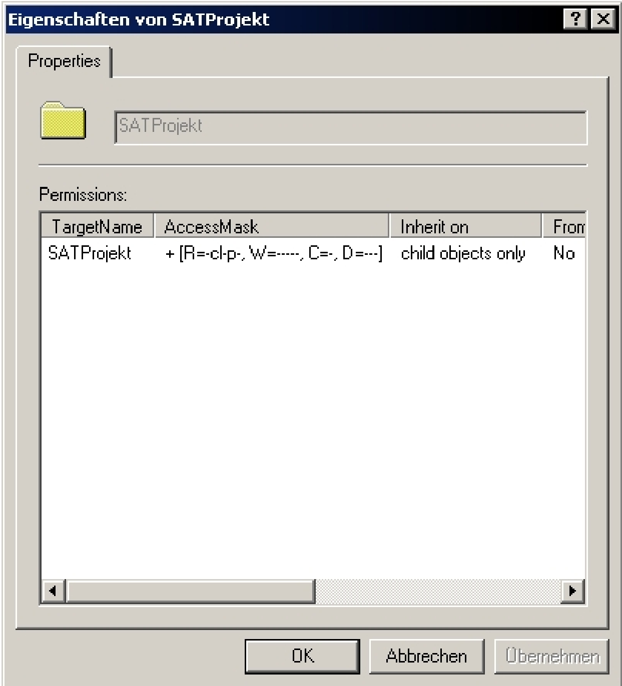
Um dies zu veranschaulichen, wird zunächst eine Bottom-Up Analyse der Gruppe „SATProjekt" (exklusive eventueller effektiven Rechte) im ADS durchgeführt. Zu diesem Zweck wird der Darstellungsmodus „target´s permissions (extended tree view)" verwendet, der Verzeichnisse zu Endknoten abstrahiert, sofern der zu untersuchende Security Principal keinerlei Rechte auf die Subknoten innehat.



Abbildungen 2.5.4.1: Extended Tree View im ADS

Abbildung 2.5.4.1 zeigt das Resultat dieser Analyse. Wie daraus ersichtlich wird, existiert in der ADS Domäne lediglich ein Verzeichnis, das Zugriffsberechtigungen für die abgefragte Gruppe aufweist. Die restlichen Verzeichnisse wurden in dieser Ansicht bereits als Endknoten dargestellt und können nicht mehr durchsucht werden. Dadurch wird eine effiziente Inspektion der potenziellen Schwachstellen ermöglicht und irrelevante Aspekte werden abstrahiert.

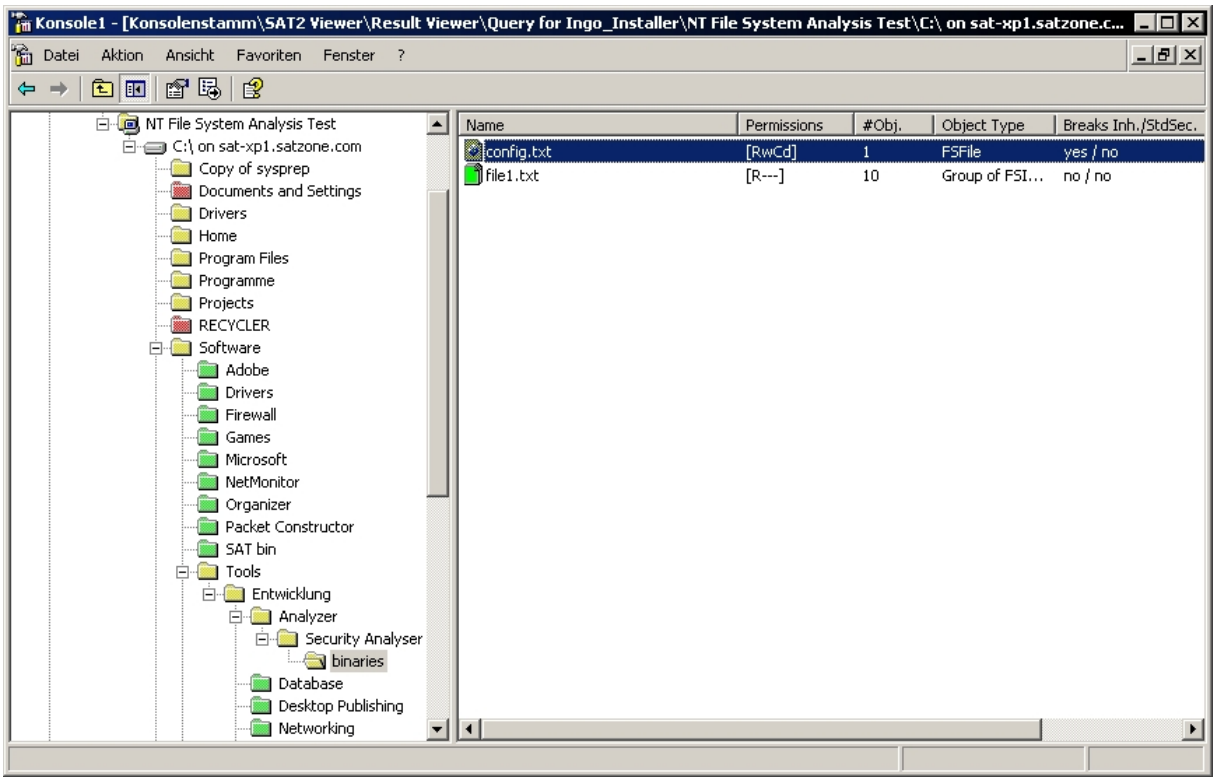
Mittels des Kontextmenüpunkts „Eigenschaften“ können nun die Einträge in der ACL des untersuchten Knotens aufgelistet werden, die zum abgebildeten Analyseergebnis und zu der daraus resultierenden Zugangsmaske geführt haben. In unserem Beispiel wurde der Knoten „SATProjekt“ grün eingefärbt, da dieser Lesezugriffsrechte für die Gruppe „SATProjekt“ auf seine Subknoten - genauer noch auf Attribute der Subknoten - vererbt (siehe Abbildung 2.5.4.2).



Abbildungen 2.5.4.2: ACE vom Knoten SATProjekt

Als abschließendes Beispiel dieses Abschnitts sollen die Unterschiede zwischen den verschiedenen Modi der Bottom-Up Analyse aufgezeigt werden. Zu diesem Zweck werden die Berechtigungen des Benutzers „Ingo_Installer“ auf den NTFS Daten des Rechners „sat-xp1.satzone.com“ inspiziert. In die folgenden Auswertungen sollen zusätzlich die effektiven Rechte aus der Mitgliedschaft in der Gruppe „SATProjekt“ mit einfließen.

Der Modus „target´s permissions (compressed extended tree view)” versucht zusätzlich zu den bereits beschriebenen Funktionen der Bottom-Up Analyse die Darstellungskompression des Viewers anzuwenden. Folglich werden Endknoten, die gleiche Zugangskontrolleinträge (ACEs) bezüglich des bzw. der analysierten SIDs aufweisen, zu einem Gruppenobjekt zusammengefasst.

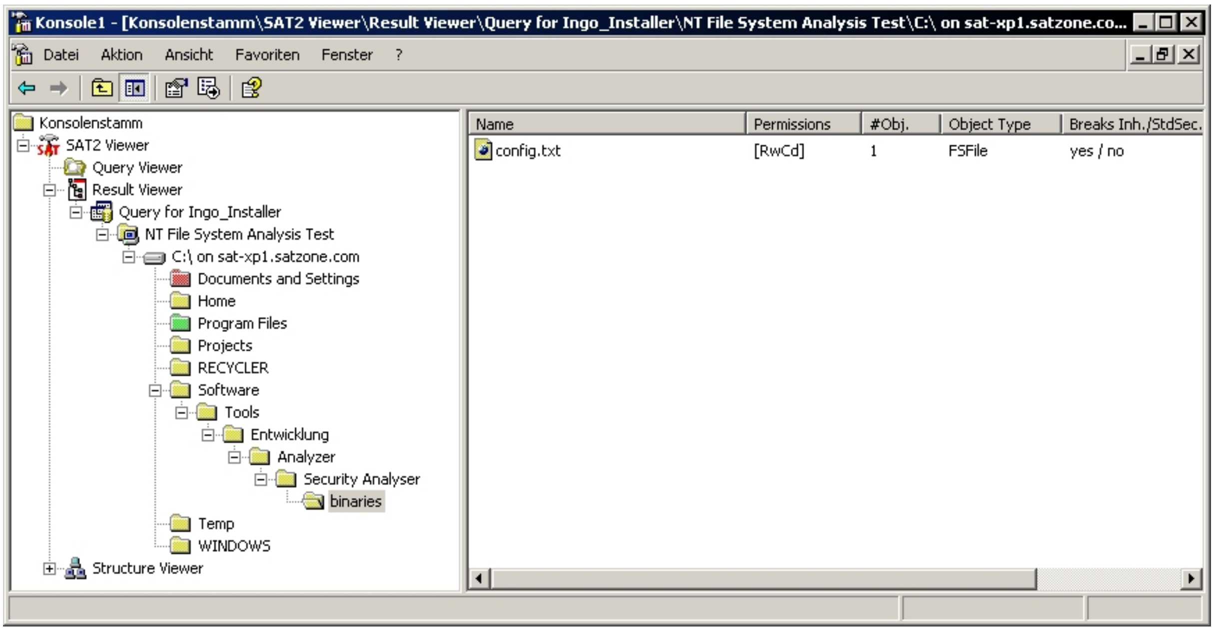


Abbildungen 2.5.4.3: Compressed Extended Tree View im NTFS

Als Ergebnis dieses Analysemodus wird dem Benutzer ein Berechtigungsbaum präsentiert (siehe Abbildung 2.5.4.3), der weitgehend die Verzeichnisstruktur des gescannten NTFS Laufwerks widerspiegelt. Dies ist darauf zurückzuführen, dass dem Benutzer „Ingo_Installer" durch einen ACE Zugriff gewährt wird, die per Default auf dem abgefragten Laufwerk weitervererbt wird. Obwohl die irrelevanten Endknoten gruppiert erscheinen, ist die Anzahl der Verzeichnisse, die vom Administrator untersucht werden müssen, dennoch unüberschaubar.

Geht man nun davon aus, dass eine gewisse Häufigkeit solcher Default Vererbungen vorliegt, und bedenkt man weiters, dass die Zweige im Berechtigungsbaum, die weder die Vererbung aufbrechen noch von der Standard Security abweichen, in der Sicherheitsanalyse ignoriert werden können, so führt dies zu dem Schluss, dass es von Vorteil ist, auch diese Zweige auszublenden.

Im Modus „target´s permissions excl. defaults (extended tree view)“ kommt genau diese Überlegung zum Einsatz. Diese Analysevariante stellt das Optimum sämtlicher vom ***SAT2*** angebotenen Modi dar.



Abbildungen 2.5.4.4: Extended Tree View excl. Defaults im NTFS

Vergleicht man das Resultat dieses Modus in Abbildung 2.5.4.4 mit dem des vorgehenden Beispiels in Abbildung 2.5.4.3, so ist augenscheinlich zu erkennen, dass sich der Verzeichnisbaum auf einen Bruchteil der Einträge reduziert hat. Speziell ist dies im Verzeichnis Software zu beobachten. In der „excl. defaults“ Variante muss nur mehr ein Zweig untersucht werden, wohingegen 10 Zweige in der Vorgängervariante aufscheinen. Zusätzlich werden in diesem Modus auch solche Endknoten nicht mehr angezeigt, die keine Ausnahme darstellen und letztendlich keinen Aufschluss über die Sicherheitslage mehr bieten können.

3 Rechtestrukturen in Windows 2000

Dieses Kapitel versucht eine kompakte Aufschlüsselung der Rechtestrukturen und der sicherheitsrelevanten Algorithmen, die in *Microsoft Windows 2000+* Betriebssystemen zum Einsatz kommen, zu erarbeiten und stellt somit die Basis für das Verständnis der Rechteanalyse des ***SAT 2.0*** dar. Daraus resultiert schließlich das Datenbankmodell, das als Ausgangspunkt für die ***SAT2*** Analysen dient.

Die Informationen, die für die Erstellung des Kapitels 3.1 verwendet wurden, stammen zum Teil aus [W2kS00] bzw. aus den Entwicklerreferenzen, die unter [Msdn04] zu finden sind.

3.1 Das Microsoft Windows 2000 Sicherheitsmodell

Dieses Kapitel bietet einen zusammengefassten Überblick über das *Microsoft Windows 2000* Sicherheitsmodell, das auch noch in späteren Versionen des Betriebssystems - *Windows XP/2003* - zur Anwendung kommt. Dabei wird vor allem Bezug auf das Autorisierungssubsystem bzw. auf die verwendeten Sicherheitsstrukturen genommen, die zum Verständnis der benutzerzentrierten Rechteanalyse von Nöten sind.

Da eine fundamentale Verbindung zwischen dem Active Directory Service und dem *Windows* Sicherheitsmodell besteht (siehe dazu [W2kS00/S.61 ff]), werden die in diesem Kapitel beschriebenen Konzepte allgemein abgehandelt. Auf die Differenzen zwischen ADS und NTFS wird in Kapitel 3.2 Bezug genommen.

3.1.1 Security Principals

Generell beschreibt der Begriff Security Principal ADS Objekte, findet weiters aber auch im *Windows 2000* Sicherheitsmodell Verwendung. Als Security Principal werden Computer, Gruppen und Benutzerkonten bezeichnet. Das Sicherheitsmodell sieht vor, diesen physikalischen Entitäten zum Erstellungszeitpunkt automatisch einen „Security Identifier“ (SID) zuzuweisen, der zur Authentifizierung und zur Zugangskontrolle zu den vorhandenen Ressourcen dient.

Die Definition eines Security Identifiers wird in [MsKb04] wie folgt beschrieben:

„Ein Security Identifier (SID) ist ein eindeutiger Wert variabler Länge, der zur Identifikation eines Security Principals … herangezogen wird.“

Ein solcher SID wird syntaktisch wie folgt konstruiert:

- S-r-x-y_1- ... -y_{n-1}-y_n

Bei den Buchstaben r, x, und y handelt es sich um Platzhalter, das S ist Präfix jedes SIDs und soll darauf hinweisen, dass es sich bei einer solchen Zeichenkette um einen Security Identifier handelt. Das r steht für die Versionsnummer der SID Struktur. Bisweilen existiert lediglich die Nummer 1, die auf die *Windows 2000* SID Struktur hinweist. Der Platzhalter x steht stellvertretend für die so genannte Identifikationsautorität. Ist x = 1 so wird von der „World Authority“ gesprochen, die auf allgemeine Gruppen wie beispielsweise auf die Gruppe „Jeder“ hinweist. Die gebräuchlichste Autorität ist die „NT Authority“, die durch x = 5 angedeutet wird. Die Platzhalterreihe y_1 bis y_{n-1} vertritt die Subautoritäten und identifiziert die Domäne, zu der der SID gehört. Beim letzten Element y_n dieser Formel spricht man vom relativen Identifizierungswert, der das eigentliche Konto bzw. die eigentliche Gruppe repräsentiert.

Zusätzlich zu den vom Betriebssystem auf diese Weise generierten SIDs existieren im *Windows* Sicherheitsmodell allgemeine Benutzerkonten und Gruppen, die unter dem Begriff „Wellknown SIDs“ zusammengefasst werden. Diese Wellknown SIDs sind konstante Zeichenketten, die in jedem *Windows* System integriert sind. Ein Überblick über die gebräuchlichsten SIDs ist in Tabelle 3.1.1.1 zu finden. Die komplette Liste inklusive der Details zu den Wellknown SIDs ist unter [MsKb04] abzurufen.

SID	Name
S-1-0-0	**Nobody**
S-1-1-0	**Everyone ***
S-1-3-0	**Creator Owner ***
S-1-3-1	**Creator Group ***
S-1-5-1	**Dialup ***
S-1-5-2	**Network ***
S-1-5-3	**Batch ***
S-1-5-4	**Interactive ***

S-1-5-6	**Service ***
S-1-5-7	**Anonymous ***
S-1-5-11	**Authenticated User ***
S-1-5-13	**Terminal Server User ***
S-1-5-18	**Local System ***
S-1-5-...-500	**Administrator**
S-1-5-...-501	**Guest**
S-1-5-...-512	**Domain Admins**
S-1-5-...-513	**Domain Users**
Built-in Groups	
S-1-5-32-544	**Administrators**
S-1-5-32-545	**Users**
S-1-5-32-546	**Guests**
S-1-5-32-547	**Power Users**

Tabelle 3.1.1.1: Liste der Wellknown SIDs

Die in Tabelle 3.1.1.1 mit * versehenen Gruppen bzw. Benutzerkonten entsprechen den 12 Wellknown SIDs, die im ***SAT2*** Viewer unter den Abfrageeinstellungen auszuwählen sind und in die Gruppenanalyse mit einbezogen werden können. Diese SIDs wurden bewusst in Anlehnung an die Funktionalität des ***SAT*** Scanners ausgewählt und sind ebenfalls in [MsKb04] genauer beschrieben.

3.1.2 Access Control Model

Die Autorisierung im *Windows* Betriebssystem unterliegt dem Access Control Model. Dieses Modell beschreibt die systeminternen Abläufe, die den autorisierten Zugriff von Security Principals auf Dateisystemressourcen regeln. Dabei hängt die Art des Zugriffs mitunter davon ab, auf welchen Objekttyp der Zugriff stattfindet. Auf Dateien beispielsweise können Leserechte, Schreibrechte, Änderungsrechte und Ausführungsrechte vergeben werden.

Um das Konzept der Zugriffskontrolle zu veranschaulichen, muss zunächst der systemnahe Vorgang, der bei einem Zugriff auf ein Objekt passiert, verdeutlicht werden. Da der Zugriff auf ein Objekt vom Benutzer ausgelöst wird, tatsächlich jedoch ein Programm bzw. genauer noch ein Thread des Programms auf ein Objekt zugreift, muss dieser Thread mit dem Benutzer assoziiert werden. Dies geschieht, indem jeder aufgerufene Thread eine Kopie des Access Tokens des Benutzers erhält. Ein solcher Access Token (siehe Abbildung 3.1.2.1) wird vom Betriebssystem generiert, sobald sich der jeweilige Benutzer am System anmeldet. Der Access Token repräsentiert den Sicherheitskontext des Benutzers, der den SID des Benutzers und die SIDs der Gruppen, denen der Benutzer angehört, beinhaltet.

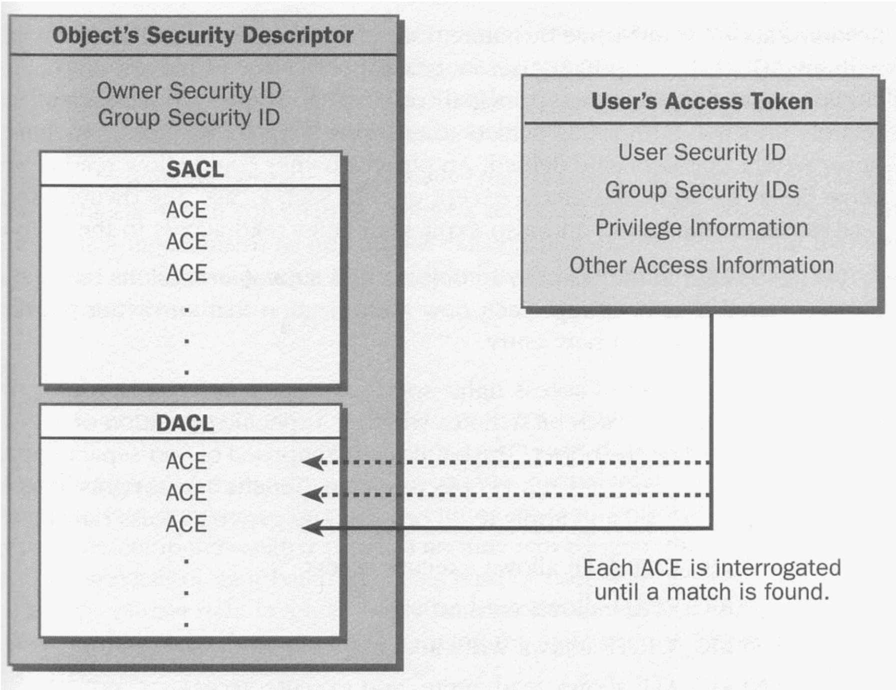


Abbildung 3.1.2.1: Access Token und Security Descriptor beim Zugriffscheck

Wird nun von einem Thread, der in diesem Zusammenhang als Agent stellvertretend für den Benutzer agiert, auf ein Objekt zugegriffen, so wird die Kopie des Access Tokens dem Betriebssystem zum Zugriffscheck vorgelegt. Dabei überprüft das Betriebssystem die Sicherheitsinformationen des jeweiligen Objekts, die in dessen Security Descriptor (siehe Abbildung 3.1.2.1) verzeichnet sind, und entscheidet daraufhin über den Zugriff.

Jede Ressource in einem *Windows 2000+* Netzwerk verfügt über einen Security Descriptor. Der Aufbau eines Security Descriptors umfasst folgende Bestandteile:

- Header
  … beinhaltet eine Revisionsnummer und eine Reihe von Control Flags, die Aussage über den Inhalt des Security Descriptors treffen.

- Owner
  In diesem Feld wird die SID des Besitzers mitgeführt.

- Primary Group
  Dieses Feld enthält die SID der primären Gruppe des Besitzers.

- Discretionary Access Control List (DACL)
  Diese Liste kann mehrere *Access Control Entries* (ACEs) enthalten, die den Zugriff auf das Objekt regeln (siehe Kapitel 3.1.3).

- System Access Control List (SACL)
  Diese Liste kann ebenfalls mehrere ACEs enthalten, die jedoch nicht der Zugriffsregelung sondern der Überwachung dienen.

Das Konzept des Besitzers, das in diesem Zusammenhang auftaucht, ist sowohl im ADS als auch im NTFS vorzufinden. Jedes Objekt verfügt über einen Besitzer. Ein Besitzer ist per Definition ein Security Principal, in den meisten Fällen handelt es sich dabei konkret um einen Benutzer. Der SID dieses Benutzers wird nun bei Objekterzeugen in den Security Descriptor des Objekts übertragen. Dadurch verfügt der Besitzer über das Recht, den Zugriff auf das Objekt, das ihm gehört, zu regeln. Der Besitz dieses Objekts bleibt solange beim Benutzer, bis das Objekt entweder gelöscht wird oder ein anderer, dazu autorisierter Benutzer den Besitz übernimmt.

3.1.3 Access Control Lists

Unter dem Begriff *Access Control List* ist eine Liste zu verstehen, die aus mehreren *Access Control Entries* (ACEs) bestehen kann und als Basis der Zugriffskontrollmechanismen in *Windows* Betriebssystemen verwendet wird. Generell ist zwischen zwei Typen von ACLs zu unterscheiden:

- Discretionary Access Control List (DACL)
- System Access Control List (SACL)

Die DACL wird dazu verwendet, um festzustellen, welcher Security Principal welche Art Zugriff auf das gewünschte Objekt erhält. Die SACL andererseits spezifiziert, welche Art Zugriff von welchem Security Principal auf das Objekt überwacht respektive mitprotokolliert werden soll. In der Sicherheitsanalyse des ***SAT 2.0*** ist die Auswertung der SACL Einträge weder vorgesehen noch im Sinne des ursprünglichen Konzepts.

Im *Windows 2000* Sicherheitsmodell ist die Verwendung von sechs verschiedenen ACE Typen in den DACLS und SACLs vorgesehen:

- Access-denied
  Der ACE Eintrag in einer DACL verweigert den Zugriff.

- Access-allowed
  Der ACE Eintrag in einer DACL erlaubt den Zugriff.

- Access-denied, objektspezifisch
  Der ACE Eintrag in einer DACL verweigert den Zugriff auf Objektattribute und beschränkt die Vererbung auf Kind-Objekte.

- Access-allowed, objektspezifisch
  Der ACE Eintrag in einer DACL erlaubt den Zugriff auf Objektattribute und beschränkt die Vererbung auf Kind-Objekte.

- System-audit
  Der ACE Eintrag in einer SACL protokolliert den Zugriff auf Objekte.

- System-audit, objektspezifisch
  Der ACE Eintrag in einer SACL protokolliert den Zugriff auf Attribute eines Objekt.

Drei dieser Typen werden als generische ACE Typen bezeichnet. Zusätzlich zu einem Flag, das den ACE Typ spezifiziert, weist die Struktur eines generischen ACE noch verschiedene Zugangskontrolldaten auf. Jeder Eintrag verfügt verständlicherweise über einen SID, der auf den Security Principal hinweist, auf den sich der ACE bezieht. Die Access Mask (siehe Kapitel 3.2), die als 32 Bit Wert im ACE aufscheint, beschreibt die eigentlichen Zugriffsrechte, die mit dem Eintrag assoziiert werden. Zusätzlich ist in der ACE Struktur eine Gruppe von Flags zu finden, die zum einen das Vererbungsverhalten regelt und zum anderen, im Falle einer SACL, die Überwachung steuert. Definierte Nachteile dieser Struktur sind einerseits die Einschränkungen in der Vererbung, da nicht bestimmt werden kann, auf welchen Objekttyp ein Recht vererbt wird. Andererseits kann der Zugriff nur für das gesamte Objekt geregelt werden.

Im Unterschied dazu stehen die objektspezifischen Typen, die ein höheres Maß an Kontrolle in Bezug auf die Vererbung bieten und im ADS Anwendung finden. Ihre Struktur setzt auf die der generischen ACE Typen auf und erweitert diese um die Felder „Object Type“ und „Inherited Object Type“. Das Feld „Object Type“ enthält einen GUID (**G**lobal **U**nique **ID**entifier) Wert, der das Subjekt der Zugriffskontrolle des objektspezifischen ACEs präzisiert. Dieser GUID kann drei verschiedene Bedeutungen haben:

- Spezifiziert der GUID den Typ eines Kind-Objekts, so kann der SID im ACE diesen Objekttyp in dem Container anlegen, auf den sich der Eintrag bezieht.

- Spezifiziert der GUID ein Attribut oder eine Menge von Attributen, so kann der SID dieses Attribut bzw. diese Attribute lesen oder schreiben.

- Spezifiziert der GUID ein erweitertes Recht, so besitzt der SID dieses Recht.

Das Feld „Inherited Object Type“ ermöglicht dem Besitzer eines Objekts genau festzulegen, welcher Typ von Kind-Objekten diesen ACE erben kann. Dieser Typ scheint dann als GUID Wert im „Inherited Object Type“ Feld auf.

In den objektspezifischen ACEs sind zwei weitere Flags inkludiert, die anzeigen, ob das „Object Type“ Feld (*ACE_OBJECT_TYPE_PRESENT*) oder das „Inherited Object Type“ Feld (*ACE_INHERITED_OBJECT_TYPE_PRESENT*) einen gültigen Wert enthält. Ist beides nicht der Fall, so ist der objektspezifische ACE wie ein generischer zu behandeln.

3.1.4 Vererbung

Das Vererbungskonzept ist charakteristischer Bestandteil des *Windows* Zugriffskontrollmodells. Per Definition ist unter Vererbung im Dateisystem der Prozess zu verstehen, bei dem Container Objekte Zugriffskontrollinformation an deren Kind-Objekte weitergeben, sowohl an Container als auch an Nichtcontainer. Dieser Prozess findet immer dann statt, wenn entweder ein neues Kind-Objekt erstellt wird oder die DACL bzw. die SACL im Security Descriptor des Vater-Containers modifiziert wird.

Eine ACL kann vererbte und explizite ACEs enthalten. Um zu differieren, um welchen ACE es sich handelt bzw. um die Vererbungsregeln des ACE zu bestimmen, sind so genannte Vererbungsflags (siehe auch Kapitel 3.1.3) im ACE definiert. Diese Flags haben in DACLs und SACLs die gleiche Bedeutung und sind wie folgt aufgeschlüsselt:

- INHERITED_ACE
  Dieses Flag zeigt an, ob der ACE vererbt wurde oder nicht.

- INHERIT_ONLY_ACE
  Dieses Flag spezifiziert, dass der ACE zwar an alle Kind-Objekte vererbt aber nicht am Objekt direkt angewendet wird.

- NO_PROPAGATE_INHERIT_ACE
  Dieses Flag bedeutet, dass der geerbte ACE nicht wiederum an Kind-Objekte weitergegeben werden kann.

- CONTAINER_INHERIT_ACE (NTFS spezifisch)
  Erbt ein Container im NTFS einen ACE, bei dem dieses Flag gesetzt wurde, so wird das INHERIT_ONLY_ACE Flag des Containers auf 0 gesetzt und der ACE wird direkt am Container angewendet.

- OBJECT_INHERIT_ACE (NTFS spezifisch)
  Erbt ein Nichtcontainer im NTFS einen ACE, bei dem dieses Flag gesetzt wurde, so wird das INHERIT_ONLY_ACE Flag des Nichtcontainers auf 0 gesetzt und der ACE wird direkt am Objekt angewendet.

- ADS_ACEFLAG_INHERIT_ACE (ADS spezifisch)
  Ist dieses Flag gesetzt, so wird der damit assoziierte ACE auf Kind-Objekte im ADS vererbt.

- ADS_ACEFLAG_VALID_INHERIT_FLAGS (ADS spezifisch)
  Dieses Flag entscheidet darüber, ob die Vererbungsregeln eines ACE im ADS angewandt werden.

Da ACEs im Zugriffscheck in strikter Reihenfolge abgearbeitet werden - vom ersten bis zum letzten Eintrag -, ist die Reihung der entscheidende Faktor. Dieses Schema unterliegt zwei simplen Regeln. Regel 1 sorgt dafür, dass „Deny" Einträge vor „Allow" Einträge gereiht werden, wodurch das Verbot Vorrang vor der Erlaubnis erhält. Die zweite Regel besagt, dass explizite ACEs, die im gebräuchlichen Umgang vom Besitzer des Objekts vergeben werden, vor vererbten ACEs gereiht werden. Dadurch können beispielsweise vererbte Rechte auf Gruppen durch ein explizites Recht auf einen einzelnen Benutzer entkräftet werden. Die vererbten ACEs werden zusätzlich noch danach gereiht, im wievielten Level die Vererbung sich bereits befindet. Aufsummiert ergibt die ACE Reihung das in Abbildung 3.1.4.1 dargestellte Schema.

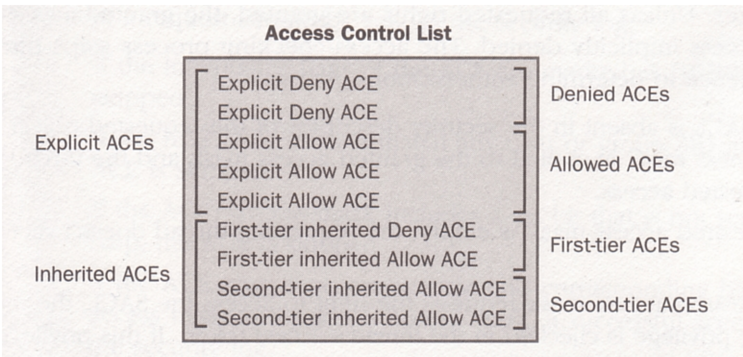


Abbildung 3.1.4.1: Reihung der ACEs in einer ACL

3.1.5 Standard Security Schema

Unter dem Standard Security Schema sind die Berechtigungen zu verstehen, die das Betriebssystem per Default auf die neu angelegten Objekte überträgt. Zu diesen Standardsicherheitseinstellungen gehören folgende Berechtigungen:

- „Local System“:
  Dieses Konto erhält Vollzugriff auf jedes neu angelegte Objekt. Handelt es sich dabei um einen Container / Ordner, so wird dieses Recht auch auf Subcontainer / Unterordner und die darin enthaltenen Dateien vererbt.

- die Gruppe „Administratoren“:
  Diese Gruppe erhält ebenfalls Vollzugriff auf jedes neu angelegte Objekt. Handelt es sich dabei um einen Container / Ordner, so wird dieses Recht wiederum auch auf Subcontainer / Unterordner und die darin enthaltenen Dateien vererbt.

- „Creator Owner“:
  Der Ersteller / Besitzer des neu angelegten Objekts erhält Vollzugriff auf dieses Objekt. Dabei wird in der ACL der Eintrag Creator Owner durch den Kontonamen des Erstellers / Besitzers ersetzt. In der Vererbung jedoch wird der Platzhalter Creator Owner auf Subcontainer / Unterordner und die darin befindlichen Dateien weitergegeben.

- Die Gruppe „Benutzer“:
  Benutzer erhalten Lese- und Ausführungsrechte auf das Objekt und auf eventuell darunter existierende Subobjekte. Sofern es sich um einen Container / Ordner handelt, dürfen Mitglieder dieser Gruppe weiters Objekte darin erstellen.

3.2 Zugriffsrechte in *Windows* und im *SAT 2.0*

Die Mechanismen, die hinter der Gewährung des Zugriffs auf ein Objekt stehen, wurden im vorigen Kapitel bereits genauer erläutert. Dieses Kapitel erörtert die einzelnen Rechte, die dabei Anwendung finden.

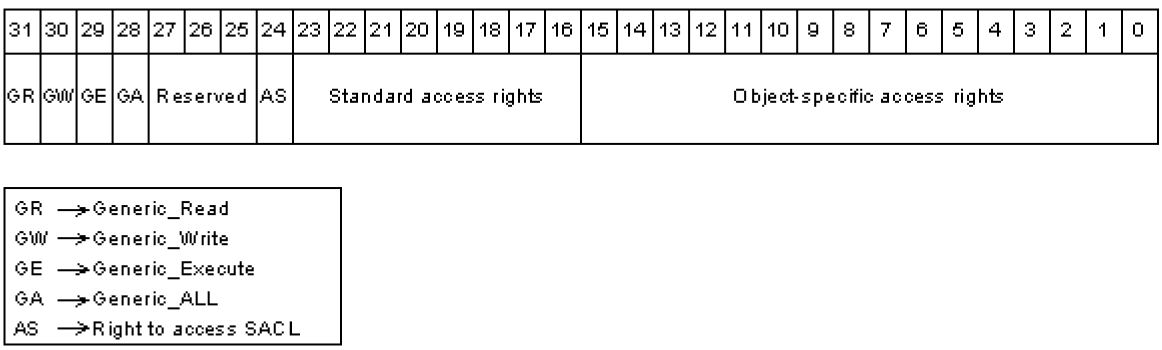


Abbildung 3.2.1: Access Mask

Wie in Kapitel 3.1.2 bzw. in Kapitel 3.1.3 schon näher thematisiert wurde, basiert die Zugriffskontrolle auf einer *Access Control List*, die mehrere *Access Control Entries* enthält.

Bestandteil eines jeden ACEs ist ein 32 Bit Wert, der den Namen Access Mask (siehe Abbildung 3.2.1) trägt und als Platzhalter für die Aktionen dient, die ein Security Principal auf einem Objekt ausführen darf.

Jedes Bit dieser Access Mask steht für eine bestimmte Operation bzw. eine Reihe von Operationen, die auf dem betroffenen Objekt durchgeführt werden können. Die Bitmaske selbst setzt sich aus vier separaten Typen von Zugriffsrechten zusammen: allgemeine, Standard, SACL und objektspezifische.

Die allgemeinen Zugriffsrechte sind schon seit *Windows NT 4.0* Bestandteil des Systems und beziehen sich auf alle Objekte. Diese Rechte finden zusätzlich auch Verwendung beim Datenverkehr mit anderen Betriebssystemen. Sie lauten wie folgt:

- GENERIC_EXECUTE (Ausführungsrecht)
- GENERIC_READ (Leserecht)
- GENERIC_WRITE (Schreibrecht)
- GENERIC_ALL (Lese-, Schreib- und Ausführungsrecht)

Aus auswertungstechnischen Gründen löst der ***SAT2*** Scanner diese Rechte in Standard und objektspezifische Rechte auf, wobei zwischen NTFS und ADS unterschieden werden muss. Die exakte Zuordnung ist in den Kapiteln 3.2.1 und 3.2.2 zu finden.

Die Standard Zugriffsrechte sind spezifischer als die allgemeinen Zugriffsrechte und sind ebenfalls auf alle Objekte anwendbar. Folgende Rechte sind darin inbegriffen:

- DELETE
  Das Recht ein Objekt zu löschen.

- READ_CONTROL
  Das Recht auf das Lesen der Berechtigungen - des Security Descriptors - eines Objekts.

- WRITE_DAC
  Das Recht auf das Ändern der Berechtigungen eines Objekts.

- WRITE_OWNER
  Das Recht die Besitzerrechte zu übernehmen.

- SYNCHRONIZE
  Das Recht auf die Synchronisation des Objekts.

Des Weiteren existiert noch das SACL Zugriffsrecht, das den Lese- bzw. Schreibzugriff auf die Überwachungseinstellungen eines Objekts regelt. Die objektspezifischen Rechte schließlich hängen vom Objekttyp ab und werden in den folgenden Kapiteln näher beschrieben.

3.2.1 Das NT File System 5

Die objektspezifischen Rechte im NTFS beziehen sich immer auf das Objekt als solches, wobei zwischen zwei Objekttypen unterschieden wird, den Containern und den Nichtcontainern. Diese Zugriffsrechte sind im Folgenden aufgelistet:

- FILE_READ_DATA / FILE_LIST_DIRECTORY
  Für Nichtcontainer stellt dieses das Recht dar, Daten zu lesen. Für Container bedeutet es das Recht, den Ordner aufzulisten.

- FILE_WRITE_DATA / FILE_ADD_FILE
  In Nichtcontainer Objekten dürfen Daten geschrieben werden. In Containern dürfen Dateien erstellt werden.

- FILE_APPEND_DATA / FILE_ADD_SUBDIRECTORY
  An Nichtcontainer Objekte dürfen Daten angehängt werden. Unter Containern dürfen Subcontainer angelegt werden.

- FILE_READ_EA
  Dieses Recht gilt für beide Objekttypen und besagt, dass erweiterte Attribute des Objekts gelesen werden dürfen.

- FILE_WRITE_EA
  Dieses Recht gilt für beide Objekttypen und besagt, dass erweiterte Attribute des Objekts geschrieben werden dürfen.

- FILE_EXECUTE / FILE_TRAVERSE
  Beschreibt das Recht, Nichtcontainer Dateien auszuführen. Im Falle eines Containers darf dieser durchsucht werden.

- FILE_DELETE_CHILD
  Dieses Recht gilt nur für Container und repräsentiert die Erlaubnis, Unterordner und Dateien zu löschen.

- FILE_READ_ATTRIBUTES
  Dieses Recht besagt, dass Attribute eines Objekts gelesen werden dürfen. Es bezieht sich auf beide Objekttypen.

- FILE_WRITE_ATTRIBUTES
  Dieses Recht besagt, dass Attribute eines Objekts geschrieben werden dürfen. Es bezieht sich auf beide Objekttypen.

- FILE_ALL_ACCESS
  Dieses Recht ist ein Aggregat sämtlicher objektspezifischen Rechte und Standard Rechte. Es bezieht sich auf beide Objekttypen wird als Vollzugriff bezeichnet.

Wie zuvor erwähnt, findet im ***SAT*** Scanner eine Auflösung der allgemeinen Rechte statt. Dabei wird eine Zuordnung durchgeführt, die auf den allgemeinen Rechten bereits im Software Development Kit vordefiniert ist:

- GENERIC_READ:
  READ_CONTROL, FILE_READ_DATA, FILE_READ_ATTRIBUTES,
  FILE_READ_EA, SYNCHRONIZE

- GENERIC_WRITE:
  READ_CONTROL, FILE_WRITE_DATA, FILE_WRITE_ATTRIBUTES,
  FILE_WRITE_EA, FILE_APPEND_DATA, SYNCHRONIZE

- GENERIC_EXECUTE:
  READ_CONTROL, FILE_EXECUTE, FILE_READ_ATTRIBUTES,
  SYNCHRONIZE

- GENERIC_ALL:
  Dieses Standard Zugriffsrecht wird auf das Recht FILE_ALL_ACCESS abgebildet.

In der Visualisierungskomponente des ***SAT*2** Systems ist man bestrebt, dem Anwender die Zugriffsrechte eines Security Principals möglichst auf einen Blick vor Augen zu führen. Da es sich jedoch um eine Vielzahl an Rechten handelt, ist eine kompakte aber dennoch aussagekräftige Darstellungsweise gefragt. Um dies zu verwirklichen, wurden die Zugriffsrechte auf einen „komprimierten" String abgebildet, wobei unter Kompression in diesem Zusammenhang lediglich eine abstrahierte Darstellung zu verstehen ist, die aus Tabelle 3.2.1.1 ersichtlich wird.

Zugriffsrecht	komprimiert	erweitert
FILE_READ_DATA	**[R---] / [r---]**	**[R=R----, W=------, C=--, D=--]**
FILE_WRITE_DATA	**[-WC-] / [-wc-]**	**[R=-----, W=W-----, C=F-, D=--]**
FILE_APPEND_DATA	**[-WC-] / [-wc-]**	**[R=-----, W=-S----, C=-D, D=--]**
FILE_READ_EA	**[R---] / [r---]**	**[R=-E---, W=------, C=--, D=--]**
FILE_WRITE_EA	**[-W--] / [-w--]**	**[R=-----, W=--E---, C=--, D=--]**
FILE_EXECUTE	**[R---] / [r---]**	**[R=--X--, W=------, C=--, D=--]**
FILE_DELETE_CHILD	**[---D] / [---d]**	**[R=-----, W=------, C=--, D=C-]**
FILE_READ_ATTRIBUTES	**[R---] / [r---]**	**[R=---A-, W=------, C=--, D=--]**
FILE_WRITE_ATTRIBUTES	**[-W--] / [-w--]**	**[R=-----, W=---A--, C=--, D=--]**
FILE_ALL_ACCESS	**[RWCD]**	**[R=REXAC, W=WSEADO, C=FD, D=CD]**
DELETE	**[---D] / [---d]**	**[R=-----, W=------, C=--, D=-D]**
READ_CONTROL	**[R---] / [r---]**	**[R=----C, W=------, C=--, D=--]**
WRITE_DAC	**[-W--] / [-w--]**	**[R=-----, W=----D-, C=--, D=--]**
WRITE_OWNER	**[-W--] / [-w--]**	**[R=-----, W=-----O, C=--, D=--]**
SYNCHRONIZE	*nicht erfasst*	*nicht erfasst*

Tabelle 3.2.1.1: NTFS Rechte Komprimierung

Der komprimierte Zugriffsrechtestring bietet die Möglichkeit einer semantischen Unterscheidung zwischen Groß- und Kleinschreibung. Wird ein Recht in großen Buchstaben dargestellt, weist dies auf das Vorhandensein sämtlicher Zugriffsrechte, die in dieser Kategorie enthalten sind, hin. Im Gegensatz dazu zeigen Kleinbuchstaben auf, dass zwar mindestens ein Recht in dieser Kategorie existiert, jedoch nicht der vollständige Satz vorhanden ist.

Die erweiterte Darstellung der Zugriffsrechte, wie sie in Tabelle 3.2.1.1 angedeutet wird, kann im ***SAT*** Viewer über den Eigenschaftsdialog (siehe Kapitel 2.4.2.2) aufgerufen werden. In diesem Dialog werden sämtliche ACEs dargestellt, die auf das Objekt gelten und mit dem analysierten Security Principal assoziiert sind.

3.2.2 Das Active Directory Service

Zusätzlich zu den Rechten, die auf ein Objekt als solches vergeben werden, erlaubt das Active Directory Service dem Besitzer die Zugriffsbeschränkung einzelner Attribute eines Objekts. Zu diesem Zweck werden folgende Zugriffsrechte herangezogen, die in Abhängigkeit von den Flags des jeweiligen ACE (siehe Kapitel 3.1.3) zu interpretieren sind:

- ADS_RIGHT_DS_CREATE_CHILD
  Das Recht ein Kind-Objekt anzulegen. Enthält das „Object Type“ Feld einen gültigen GUID, so dürfen nur Objekte dieses Typs angelegt werden.

- ADS_RIGHT_DS_DELETE_CHILD
  Das Recht ein Kind-Objekt zu löschen. Enthält das „Object Type“ Feld einen gültigen GUID, so dürfen nur Objekte dieses Typs gelöscht werden.

- ADS_RIGHT_ACTRL_DS_LIST
  Das Recht Kind-Objekte aufzulisten.

- ADS_RIGHT_DS_SELF
  Das Recht eine Schreiboperation durchzuführen. Enthält das „Object Type" Feld einen gültigen GUID, so wird das Schreibrecht auf diesen konkreten Objekttyp eingeschränkt.

- ADS_RIGHT_DS_READ_PROP
  Das Recht Attribute des Objekts zu lesen. Enthält das „Object Type“ Feld einen gültigen GUID, so dürfen nur diese Attribute gelesen werden.

- ADS_RIGHT_DS_WRITE_PROP
  Das Recht Attribute des Objekts zu schreiben. Enthält das „Object Type“ Feld einen gültigen GUID, so dürfen nur diese Attribute geschrieben werden.

- ADS_RIGHT_DS_DELETE_TREE
  Dieses Recht gilt nur für Container und repräsentiert die Erlaubnis, sämtliche Kind-Objekte zu löschen, ohne auf dessen Zugriffsrechte Rücksicht zu nehmen.

- ADS_RIGHT_DS_LIST_OBJECT
  Das Recht ein bestimmtes Objekt aufzulisten.

- ADS_RIGHT_DS_CONTROL_ACCESS
  Das Recht eine Operation auszuführen, die von einem erweiterten Zugriffsrecht kontrolliert wird. Enthält das „Object Type“ Feld einen gültigen GUID, so deutet dies auf ein konkretes erweitertes Recht hin. Ist das Feld leer, so dürfen auf dem Objekt alle Operation, die von erweiterten Rechten kontrolliert werden, ausgeführt werden.

Die Auflösung der allgemeinen Rechte wird im ADS ebenfalls durchgeführt. Die Zuordnung passiert folgendermaßen:

- GENERIC_READ:
  READ_CONTROL, ADS_RIGHT_DS_LIST_OBJECT,
  ADS_RIGHT_DS_READ_PROP, ADS_RIGHT_ACTRL_DS_LIST

- GENERIC_WRITE:
  READ_CONTROL, ADS_RIGHT_DS_WRITE_PROP, ADS_RIGHT_DS_SELF

- GENERIC_EXECUTE:
  READ_CONTROL, ADS_RIGHT_ACTRL_DS_LIST

- GENERIC_ALL:
  ADS_RIGHT_DS_CREATE_CHILD, ADS_RIGHT_DS_DELETE_CHILD, ADS_RIGHT_ACTRL_DS_LIST, ADS_RIGHT_DS_SELF, ADS_RIGHT_DS_READ_PROP, ADS_RIGHT_DS_WRITE_PROP, ADS_RIGHT_DS_DELETE_TREE, ADS_RIGHT_DS_LIST_OBJECT, ADS_RIGHT_DS_CONTROL_ACCESS, DELETE, READ_CONTROL, WRITE_DAC, WRITE_OWNER

Analog zur Darstellung der Rechte im NTFS werden auch die ADS Zugriffsrechte auf zwei Varianten dargestellt (siehe Tabelle 3.2.2.1). Der komprimierte Zugriffsrechtestring soll wiederum einen übersichtlichen, ersten Eindruck im Result Pane des ***SAT2*** Viewer Snap-Ins präsentieren. Die erweiterte Variante, die über den Eigenschaftsdialog (siehe Kapitel 2.4.2.2) ersichtlich wird, gibt konkreten Aufschluss über die exakten Rechte in den ausgelesenen ACEs.

Zugriffsrecht	komprimiert	erweitert
ADS_RIGHT_DS_CREATE_CHILD	**[--C-] / [--c-]**	**[R=------, W=-----, C=C, D=---]**
ADS_RIGHT_DS_DELETE_CHILD	**[---D] / [---d]**	**[R=------, W=-----, C=-, D=-O-]**
ADS_RIGHT_ACTRL_DS_LIST	**[R---] / [r---]**	**[R=--L---, W=-----, C=-, D=---]**
ADS_RIGHT_DS_SELF	**[RW--] / [rw--]**	**[R=-----S, W=----S, C=-, D=---]**
ADS_RIGHT_DS_READ_PROP	**[R---] / [r---]**	**[R=----P-, W=-----, C=-, D=---]**
ADS_RIGHT_DS_WRITE_PROP	**[-W--] / [-w--]**	**[R=------, W=---P-, C=-, D=---]**
ADS_RIGHT_DS_DELETE_TREE	**[---D] / [---d]**	**[R=------, W=-----, C=-, D=--T]**
ADS_RIGHT_DS_LIST_OBJECT	**[R---] / [r---]**	**[R=---O--, W=-----, C=-, D=---]**
ADS_RIGHT_DS_CONTROL_ACCESS	**[RW--] / [rw--]**	**[R=A-----, W=A----, C=-, D=---]**
DELETE	**[---D] / [---d]**	**[R=------, W=-----, C=-, D=C--]**
READ_CONTROL	**[R---] / [r---]**	**[R=-C----, W=-----, C=-, D=---]**
WRITE_DAC	**[-W--] / [-w--]**	**[R=------ W=-D---, C=-, D=---]**
WRITE_OWNER	**[-W--] / [-w--]**	**[R=------, W=--O--, C=--, D=--]**
SYNCHRONIZE	*nicht erfasst*	*nicht erfasst*

Tabelle 3.2.2.1: ADS Rechte Komprimierung

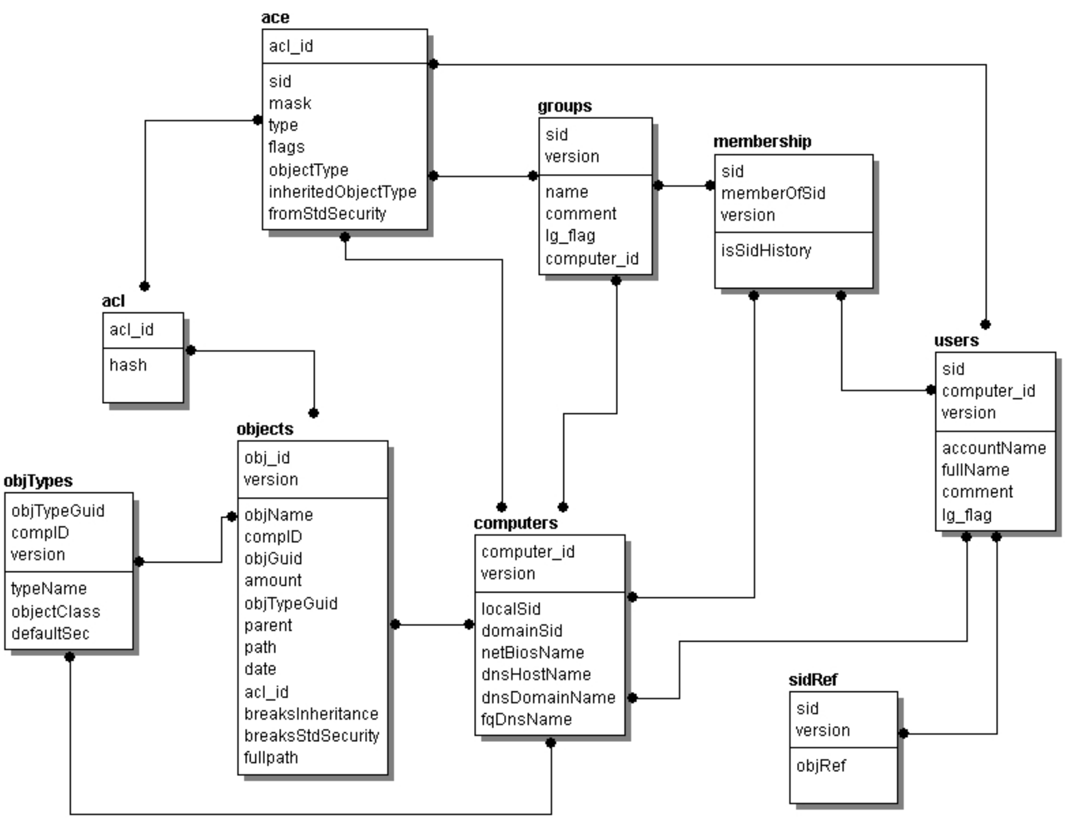
Die in beiden Darstellungen verwendete Groß-/Kleinschreibung differiert jedoch in der Interpretation von den Darstellungen im NTFS. Großbuchstaben deuten hier darauf hin, dass alle Zugriffsrechte für das gesamte Objekt zu verstehen sind, wohingegen Kleinbuchstaben verdeutlichen sollen, dass es sich um ein Zugriffsrecht auf ein Attribut des Objekts handelt.

Zusätzlich zu den „normalen“ Zugriffsrechten existieren im Active Directory so genannte erweiterte Rechte („extended Rights“), die ihrerseits durch das Recht

„ADS_RIGHT_DS_CONTROL_ACCESS“ kontrolliert werden. Diese erweiterten Rechte befinden sind im Active Directory in einem eigenen Container und beinhalten zwei Attribute. Zum einen wird ein GUID mitgeführt, der die Objekttypen referenziert, für die dieses erweiterte Recht anwendbar sein soll. Zum anderen ist ein weiterer GUID enthalten, der das eigentliche Recht identifizieren soll. Mittels dieses Mechanismus wird dem Anwender ermöglicht, eigene Rechte auf sein Objekt zu definieren. Dadurch wird die Flexibilität der Zugriffsrechte immens erweitert. An dieser Stelle sei erwähnt, dass das ***SAT2.0*** System diese erweiterten Rechte bisweilen nicht unterstützt.

3.3 Datenbankschema

Dieses Kapitel erläutert die einzelnen Tabellen der Datenbank und die darin enthaltenen Felder. Zur Übersicht über das Datenbankkonzept soll das ER-Diagramm in Abbildung 3.3.1 dienen. Die in den folgenden Tabellen fett gedruckten Einträge weisen auf den Primärschlüssel der Tabelle hin.



Abbildungen 3.3.1: Entity-Relationship-Modell der SAT2 Datenbank

Beim Datenbankmodell wurde einstweilen aus Gründen der Systemarchitektur auf größere Performanzoptimierung und Normalisierung weitgehend verzichtet. Da das Modell ursprünglich entworfen wurde um die Daten, die aus den einzelnen Scanroutinen hervorgehen, möglichst in systemnahem Zustand abzulegen, und da weiters diverse Aspekte des Betriebssystems in den Datenbestand mit einfließen, wurden Designrichtlinien großteils außer Acht gelassen. Es versteht sich, dass das Modell durchaus optimiert werden kann. Dies würde jedoch einer Umprogrammierung sämtlicher Komponenten des ***SAT2*** bedürfen.

3.3.1 Die „ace“ Tabelle

Diese Tabelle sammelt sämtliche ACEs, die im Security Scan ausgelesen werden. Dabei repräsentiert ein Eintrag in dieser Tabelle sämtliche Daten eines ACE-Headers, wie sie in Kapitel 3.1.3 erläutert sind.

Attribut	Datentyp	Beschreibung
acl_id	INT	Referenz auf die ACL, dem dieser ACE zugeordnet ist
sid	VARCHAR	SID eines Security Principals
mask	TINYINT	Access Mask
type	TINYINT	ACE Typ
flags	TINYINT	Vererbungsflags
objType	VARCHAR	Object Type GUID
inheritedObjectType	VARCHAR	Inherited Object Type GUID
fromStdSecurity	BIT	Flag zur Unterscheidung ob ACE zum Standard Security Schema gehört

Tabelle 3.3.1.1: Definition von ACE

3.3.2 Die „acl“ Tabelle

In dieser Tabelle werden die verschiedenen ACLs referenziert. Obwohl jedes Objekt eine ACL beinhaltet, existieren viele ACLs mit den gleichen ACEs. Dies wird hier ausgenutzt. Mittels des „hash“ Eintrags wird versucht, die Anzahl der Tabelleneinträge drastisch zu reduzieren und die hohe Redundanz in den verschiedenen ACLs zu umgehen. Zur Berechnung dieses Hash-Werts werden die „access mask“ Einträge der ACEs einfach aufsummiert.

Attribut	Datentyp	Beschreibung
acl_id	INT	ID Wert der ACL
hash	INT	Hash-Wert, der die ACEs in der ACL repräsentiert

Tabelle 3.3.2.1: Definition von ACL

3.3.3 Die „computers“ Tabelle

Hier werden Informationen über sämtliche Computer im Netz abgelegt, die zur Referenz der Objektherkunft herangezogen werden und auch als Security Principal auftreten können.

Attribut	Datentyp	Beschreibung
computer_id	INT	ID des Computers
localSid	VARCHAR	lokaler SID des Computers
domainSid	VARCHAR	Domain SID des Computers
netBiosName	VARCHAR	NetBIOS Name des Computers
dnsHostName	VARCHAR	DNS Name des Computers
dnsDomainName	VARCHAR	Referenz auf die Domäne, in der sich der Computer befindet
fqDnsName	VARCHAR	Fully quallified DNS Name
version	INT	Scan Version

Tabelle 3.3.3.1: Definition von computers

3.3.4 Die „groups“ Tabelle

In dieser Tabelle sind Informationen über die Gruppen verzeichnet, die auf den einzelnen Rechnern lokal bzw. in der Domäne existieren.

Attribut	Datentyp	Beschreibung
sid	VARCHAR	Gruppen SID
name	VARCHAR	Name der Gruppe
comment	VARCHAR	Kommentar
lg_flag	VARCHAR	Flag, das zwischen lokaler und Domänengruppe unterscheidet
computer_id	INT	Referenz auf den Computer, auf dem die Gruppe ausgelesen wurde
version	INT	Scan Version

Tabelle 3.3.4.1: Definition von groups

3.3.5 Die „membership“ Tabelle

Diese Tabelle verzeichnet Mitgliedschaftsbeziehungen zwischen Benutzern, Computern und Gruppen. Der Eintrag „isSidHistory“ bezieht sich auf einen Sonderfall im ADS, indem für einen Security Principal mehrere SIDs vorliegen (Kompatibilität zu *Windows NT 4*).

Attribut	Datentyp	Beschreibung
sid	VARCHAR	SID des jeweiligen Security Principals
memberOfSid	VARCHAR	SID der Gruppe, in der der Security Principal Mitglied ist

isSidHistory	BIT	SID stammt aus der SID History
version	INT	Versionsnummer des Security-Scans

Tabelle 3.3.5.1: Definition von membership

3.3.6 Die „objects“ Tabelle

Diese Tabelle vereint sämtliche Objekte, die in der ADS Analyse sowie in der NTFS Analyse ausgelesen werden. Die Einträge „breaksStdSecurity“ und „breaksInheritance“ wurden bereits in Kapitel 2.4.2.2 erörtert. Wichtig in dieser Tabelle ist weiters der Eintrag „path“. Durch diese „Pfadangabe“, die den tatsächlichen Verzeichnispfad auf einen String abgebildet wiedergibt, wird es ermöglicht, eine effiziente Bottom-Up Analyse durchzuführen.

Attribut	Datentyp	Beschreibung
obj_id	INT	ID des Objekts
objName	VARCHAR	der Objektname
objGuid	VARCHAR	GUID des Objekts
comp_id	INT	Referenz auf den Computer, auf dem das Objekt ausgelesen wurde
acl_id	INT	Referenz auf die ACL des Objekts
objTypeGuid	VARCHAR	Objekttyp GUID (ADS)
amount	INT	repräsentiert die Anzahl der in diesem Eintrag zusammengefassten Objekte (bei Kompression)
version	INT	Scan Version
parent	INT	Referenz auf den Vaterknoten
path	VARCHAR	Stringabbildung der Verzeichnisstruktur über dem Objekt
breaksStdSecurity	BIT	Flag, ob Standard Security Schema verlassen wurde
breaksInheritance	BIT	Flag ob Vererbung unterbrochen wurde
date	CHAR	Erstellungs- bzw. letztes Modifikationsdatum des Objekts
fullPath	VARCHAR	Pfadangabe zum Objekt

Tabelle 3.3.6.1: Definition von objects

3.3.7 Die „objTypes“ Tabelle

Diese Tabelle ermöglicht das Identifizieren der GUID Verweise auf einen Objekttyp. Da NTFS lediglich zwei Objekttypen, Verzeichnisse und Dateien, unterstützt, wird diese Tabelle nur in der ADS Analyse beschrieben.

Attribut	Datentyp	Beschreibung
objTypeGuid	VARCHAR	GUID des Objekttyps
compID	INT	Referenz auf den Computer, auf dem der Objekttyp ausgelesen wurde
TypeName	VARCHAR	Objekttypbezeichnung
objectClass	CHAR	Objektklasse
defaultSec	INT	Flag ob Objektklasse der Std. Security entspricht
version	INT	Scan Version

Tabelle 3.3.7.1: Definition von objTypes

3.3.8 Die „sidRef“ Tabelle

In dieser Tabelle werden alle Objekte des Active Directory mit einer Referenz auf das Vaterobjekt gespeichert, um nachvollziehen zu können, wo ein Security Principal im ADS angelegt wurde.

Attribut	Datentyp	Beschreibung
sid	VARCHAR	SID des Objektes (im ADS)
objRef	INT	SID des zugehörigen Vaterknotens
version	INT	Scan Version

Tabelle 3.3.8.1: Definition von sidRef

3.3.9 Die „users“ Tabelle

Diese Tabelle verzeichnet Informationen über sämtliche Benutzerkonten im Netz, die wahlweise auf den einzelnen Rechnern lokal oder in der Domäne ausgelesen werden.

Attribut	Datentyp	Beschreibung
sid	VARCHAR	SID des Benutzerkontos
accountName	VARCHAR	Kontoname
fullName	VARCHAR	Vollständiger Benutzername
comment	VARCHAR	Kommentar
lg_flag	VARCHAR	Flag, das zwischen lokalem und Domänenbenutzer unterscheidet
computer_id	INT	Referenz auf den Computer, auf dem die Gruppe ausgelesen wurde
version	INT	Scan Version

Tabelle 3.3.9.1: Definition von users

4 Leitfaden für MMC Entwickler

Dieses Kapitel beschreibt ausführlich die Microsoft Management Console (MMC) und die Realisierung eines Softwareprojekts, das sich der MMC als Framework bedient. Dabei wird erklärt, wie man ein so genanntes Snap-In für die MMC unter zu Hilfenahme der Active Template Library (ATL) 3.0 in der Programmiersprache C++ entwickelt. Anhand von Beispielen aus den MMC Komponenten des ***SAT*** Projektes wird gezeigt, auf welche Implementierungsdetails man achten muss, um ein MMC Projekt effizient realisieren zu können.

Die in diesem Kapitel beschriebenen Vorgehensweisen repräsentieren die Erfahrungen des Autors dieser Arbeit im Umgang mit der MMC Programmierung in C++ und resultieren größtenteils aus dem Studium der Dokumentation des Microsoft Platform Software Development Kit [MsLP04]. Die Artikel [Bol00] und [CoPr01] gaben hierfür den größten Aufschluss. Die in Kapitel 4.1 beschriebenen allgemeinen Daten zur Microsoft Management Console wurden aus [Mmc00] und [Mwhp99] sinngemäß übernommen.

4.1 Die Microsoft Management Console

Unter die Aufgaben eines Netzwerkadministrators fällt eine Vielzahl von Problemstellungen und Anforderungen. Dabei muss es ihm möglich sein, sein Netzwerk und alle darin enthaltenen Komponenten effizient und im Idealfall zentral verwalten zu können. Dies bringt eine Reihe von Problemen mit sich, die im Folgenden kurz erwähnt sein sollen:

- ***Individuelles Benutzerinterface der verwendeten Werkzeuge***
  Im alltäglichen Server- bzw. Netzwerkbetrieb kommen typischerweise mehrere Administrationsanwendungen zum Einsatz. Diese betreffen Netzwerkressourcen, Datenbanken, Serverdienste, etc. und sind je nach Hersteller verschieden zu bedienen bzw. präsentieren sich in unterschiedlichsten Benutzerinterfaces.

- ***Fehlende oder unzureichende Adaptionsmöglichkeiten***
  Die meisten Administrationsanwendungen setzen gewisse, oft herstellerspezifische Grundkenntnisse voraus und sind auf einen bestimmten Ablauf hin fixiert. Dies stellt

sowohl für erfahrene als auch für unversierte Benutzer eine Hürde dar, die sie entweder unnötig aufhält oder die Arbeit verkompliziert.

- ***Der Überblick über die Verwaltungswerkzeuge geht verloren***
  In modernen Netzwerkbetriebssystemen gibt es eine Reihe von Programmen, die ähnliche Funktionalität aufweisen oder von Systemversion zu Systemversion ihr Bedienverhalten ändern. Weiters wird im Internet eine Unzahl von Anwendungen angeboten, die teilweise die gesuchte Funktionalität bieten, jedoch den Anforderungen selten vollständig entsprechen. In Folge dessen geht der Überblick über die „Werkzeugsammlung" leicht verloren.

- ***Zentralisierung der Netzwerkverwaltung wird selten unterstützt***
  Die meisten Administrationsanwendungen sind darauf ausgelegt, dass sie am Rechner vor Ort installiert, gestartet und ausgewertet werden. Dies zwingt den Administrator dazu, jeden Rechner einzeln anstatt alle parallel zu betreuen.

Von diesen Überlegungen inspiriert fand sich die Firma *Microsoft* veranlasst, verbesserte Werkzeuge zur Administration von *Windows*-basierten Systemen zu entwickeln. In folge dessen wurde ein Projektziel formuliert, das auf die Kernanforderungen der Endbenutzer eingehend reagieren sollte. Das MMC Projekt versucht administrative Aufgaben mittels konsequenter Integration der Verwaltungswerkzeuge, strikter Ausrichtung auf die eigentlichen Kernaufgaben und Vereinfachung der Interfaceschnittstellen zu erleichtern. Als *Microsoft* dieses Ziel spezifizierte, wurde das Projekt zum essentiellen Bestandteil der *Windows* Verwaltungsmanagement Strategie und wurde folglich zum Framework für alle Administrationsanwendungen, die in *Windows* inkludiert sind.

Die erste Version der Microsoft Management Console (1.0) wurde bereits mit *Windows NT 4* ausgeliefert und war Bestandteil der *Option Pack* für *Windows NT Server* Betriebssysteme. MMC 1.1 stellte den nächsten Entwicklungsschritt dar und war im *Microsoft SQL Server 7.0* und im *Microsoft Systems Management Server 2.0* integriert. Erstmals allgemeine Verbreitung fand die Management Console mit dem Betriebssystem *Windows 2000* und ist seit dieser Version sowohl auf allen 32- als auch auf den 64-Bit Windows Systemen lauffähig. Die zum Zeitpunkt der Fertigstellung dieser Arbeit aktuellste Version 2.0 der MMC ist in *MS Windows XP* und der *Windows 2003* Serverfamilie enthalten und ist abwärtskompatibel zur Version 1.2.

Aufgrund der Tatsache, dass mit Einführung der Version 1.2 auch die Active Directory Technologie verstärkt propagiert wurde und da zu Beginn des ***SAT2*** Projektes die MMC Version 2.0 noch nicht verfügbar war, entschied sich das Projektteam zur Verwendung der Version 1.2.

Grundsätzlich bildet die MMC eine Kernkomponente in der *Windows*-internen Verwaltungsinfrastruktur, die auf den so genannten Windows Management Services (siehe Abbildung 4.1.1) basiert. Diese Management Services sind wiederum generell in 3 logische Schichten aufteilbar:

- ***Common Services***
  In diese Schicht fallen sämtliche low-level Betriebssystemdienste und sie bildet somit die Basis der Windows Management Services. Hier sind beispielsweise Dienste wie das Active Directory Service, der Event Notification Dienst und die Windows Management Instrumentations zu nennen.

- ***Management Logic***
  Diese Zwischenschicht bietet 2 verschiedene Arten von Service-Klassen an. In die erste Service-Klasse fallen Standard Management Werkzeuge, die auf den Common Services aufsetzen, wie zum Beispiel der Group Policy Editor. Die 2. Klasse beschreibt den Bereich der nicht im Betriebssystemumfang enthaltenen Management Lösungen bzw. Erweiterungen, in der unter anderem auch die Hauptkomponenten von ***SAT2*** anzusiedeln wären.

- ***Presentation***
  In dieser Schicht sind die high-level Dienste zusammengefasst, die dem Benutzer die übersichtliche und einfache Handhabung der Dienste aus den beiden anderen Schichten ermöglichen. Die MMC ist teil dieser Schicht.

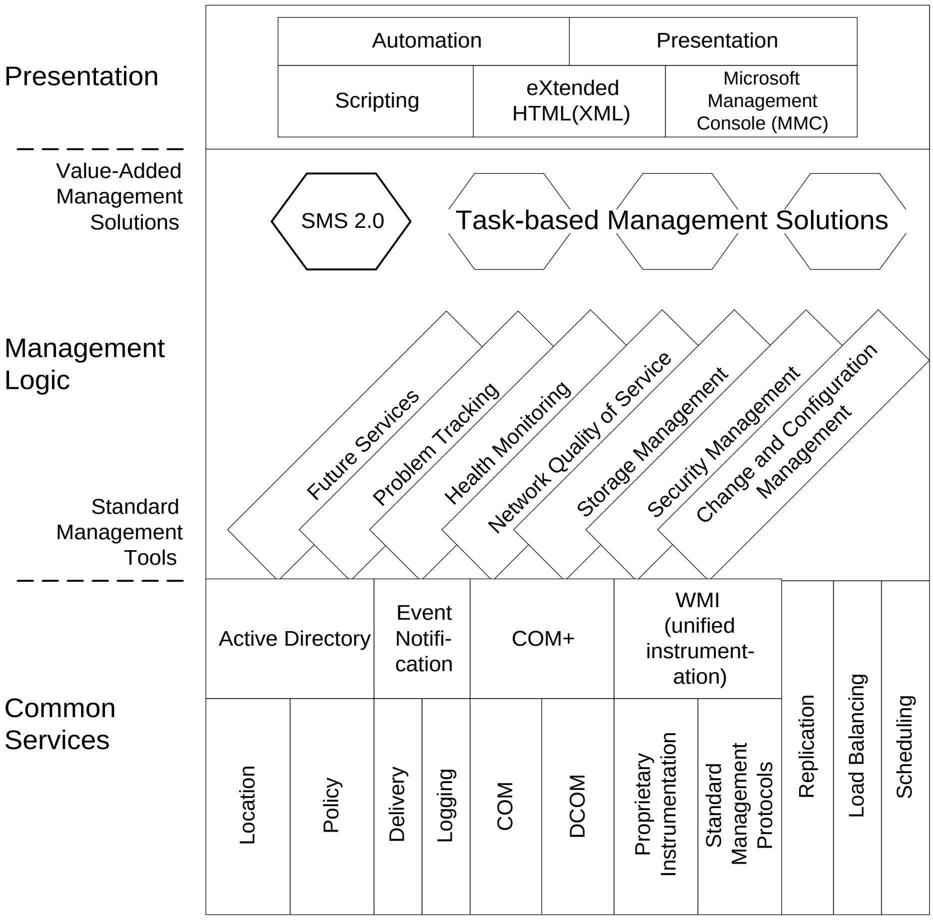


Abbildung 4.1.1: Windows Management Services

Im Konkreten ist die Microsoft Management Console eine erweiterbarere Präsentationsoberfläche für Verwaltungsanwendungen, und bildet seit *Windows 2000* einen integralen Bestandteil der mitgelieferten Computeradministrationswerkzeuge. Sie stellt ein allgemeines Rahmensystem für so genannte Snap-Ins zur Verfügung, die wiederum die eigentliche Funktionalität der jeweiligen Verwaltungs- bzw. Administrationssoftware enthalten. Das der MMC zugrunde liegende Modell ermöglicht seinem Anwender beliebige Snap-Ins – in gewissem Sinne einzelne Werkzeuge – in einer Konsolenanwendung zu kombinieren (siehe Abbildung 4.1.2) und als .MSC Dateien abzuspeichern. Somit wird dem Administrator die Möglichkeit offeriert, seine Verwaltungsanwendungen adaptiv auf seine Aufgaben auszurichten und in einer einzigen MMC Konsole – in einem „Werkzeugkasten", um im Bild zu bleiben – zu sammeln.

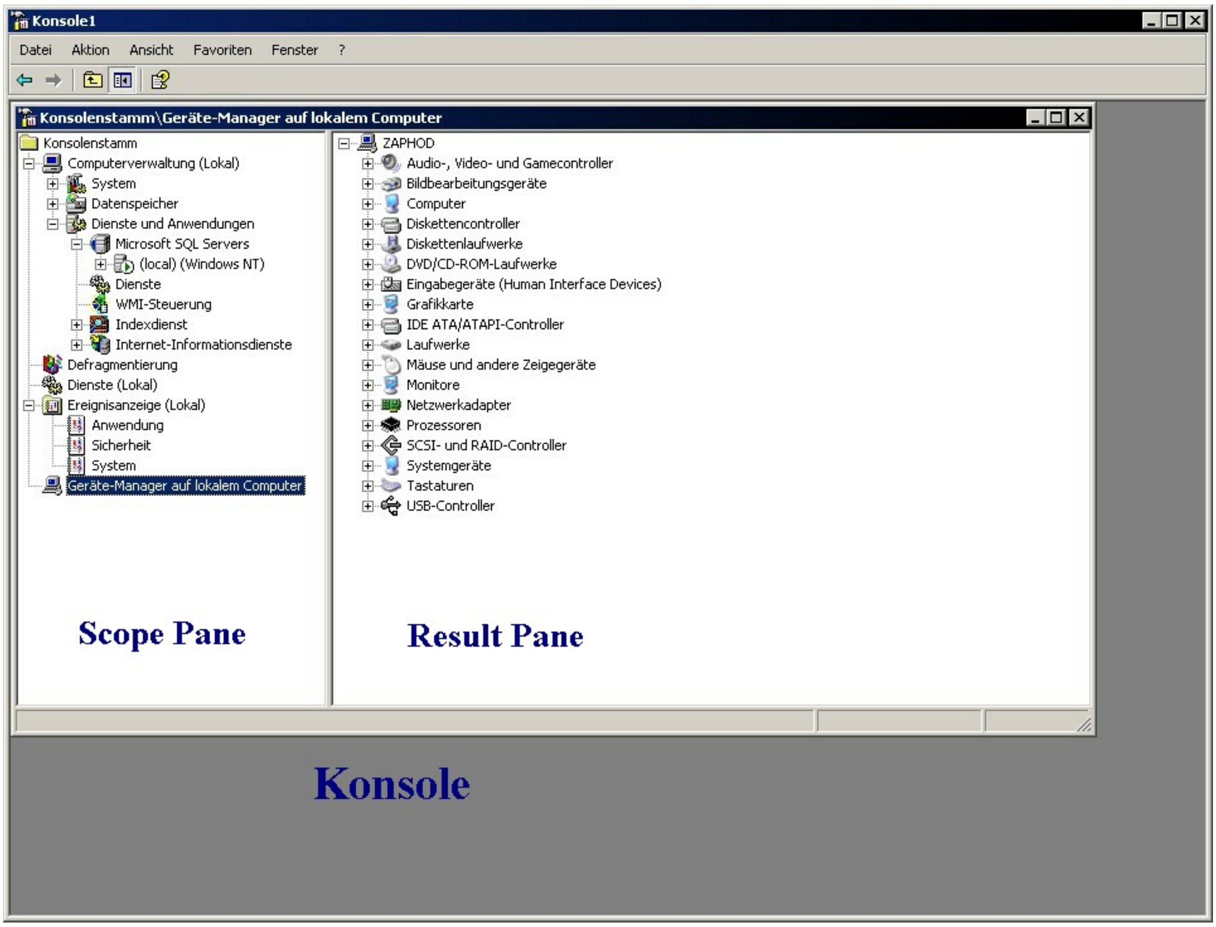


Abbildung 4.1.2: MMC Konsole mit Snap-Ins

Der Aufbau eines MMC Snap-Ins erinnert an die Baumstruktur des Windows Explorer, die als Navigationsinstanz dient und in der linken Seite der Konsole, dem Scope Pane, vorzufinden ist. Das Snap-In wird durch einen Wurzelknoten repräsentiert, der eine beliebige Anzahl von Söhnen haben kann. Die Tiefe dieses Baumes ist theoretisch unbegrenzt. Somit kann jeder Knoten als Vaterknoten fungieren und wiederum beliebig viele Söhne haben.

Wählt ein Benutzer einen Knoten im Scope Pane aus, so wird die Repräsentation des Knotens im Result Pane dargestellt, wobei folgende Darstellungsarten möglich sind:

- **Listen-Darstellung:**
  Die Listenansicht stellt die gängigste Variante der Darstellungsarten im Result Pane dar und ist eine Auflistung der Söhne des selektierten Knotens. Ähnlich dem Windows Explorer hat man auch hier die Möglichkeit, die Darstellung zu variieren, d.h. die Anzeige großer Symbole, kleiner Symbole oder Details ist möglich. Die Detail-Ansicht

offeriert zusätzlich die Option, verschiedene Attribute der im Result Pane dargestellten Knoten ein- bzw. auszublenden.

- **Taskpad-Darstellung:**
  Unter der Taskpadansicht ist die grafische Darstellung der Funktionen eines Knotens zu verstehen. Dies kann nützlich sein, um Benutzern zu erledigende Konfigurationen aufzuführen oder mögliche Programmabläufe zu schildern.

- **ActiveXControl-Darstellung:**
  Ein Snap-In hat die Möglichkeit, benutzerdefinierte **OLE** (**O**bject **L**inking and **E**mbedding) Steuerelemente im Result Pane anzuzeigen. Diese werden auch als ActiveX Controls bezeichnet und gewähren Zugriff auf unterschiedlichste Datenobjekte, die in angemessener Weise dargestellt werden.

- **Web-Darstellung:**
  Die MMC bietet eine Darstellung von HTML-Seiten an, die wahlweise lokal gespeichert sind oder mittels einer URL angegeben werden können. Diese Funktion sollte jedoch nicht als Web-Browser verstanden werden.

- **Message-Darstellung:**
  Diese Ansicht kann verwendet werden, um den Benutzer Fehlermeldungen anzuzeigen oder Informationen über den Programmablauf zu geben.

Um mit den verschiedenen Anwendungs- und Darstellungsmöglichkeiten der MMC vertraut zu werden, empfiehlt es sich, die von Microsoft entwickelten Snap-Ins zu begutachten. Im Lieferumfang von *Windows NT/2000/XP/2003* gibt es eine Reihe von Standard Verwaltungskonsolen für die MMC, die im „SYSTEM32“ Verzeichnis zu finden sind und die in nachfolgender Tabelle (Tabelle 4.1.1) kurz aufgelistet werden.

Dateiname	Funktionalität
certmgr.msc	Verwaltung der Zertifikate für Benutzer, Computer oder Dienste
ciadv.msc	Indexdienstübersicht
compmgmt.msc	Computerverwaltung
devmgmt.msc	Gerätemanager
dfrg.msc	Defragmentierung

diskmgmt.msc	Datenträgerverwaltung
eventvwr.msc	Ereignisanzeige
fsmgmt.msc	Verwaltung der Ordnerfreigaben
gpedit.msc	Group Policy Editor
lusrmgr.msc	Verwaltung von lokalen Benutzern und Gruppen
ntmsmgr.msc	Wechselmedienverwaltung
ntmsoprq.msc	Operatoranforderungen für Wechselmedien
perfmon.msc	Systemleistungsmonitor
rsop.msc	Verwaltung von Richtlinienergebnissätzen
secpol.msc	Verwaltung der lokalen Sicherheitseinstellungen
services.msc	Diensteverwaltung
wmimgmt.msc	WMI-Steuerung, Windows-Verwaltungsinstrumentation
com\comexp.msc	Verwaltung der Komponentendienste

Tabelle 4.1.1.: Übersicht Verwaltungskonsolen

4.2 Erstellen eines Snap-Ins

Ein MMC Snap-In ist eine COM in-process Serverapplikation. Per Definition bedeutet dies, dass es sich dabei um eine DLL Datei handelt, die COM-basierte Komponenten enthält. Diese Serveranwendungen werden direkt in den Adressraum des Clients geladen und benutzen keinen eigenen Adressraum, sondern den Adressraum des Clients. Um eine Snap-In Anwendung laden zu können, muss diese in der Windows Registry unter folgendem Schlüssel registriert werden:

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

Zur Realisierung der MMC Komponenten des ***SAT2*** Projektes wurde die Entwicklungsumgebung *Microsoft Visual Studio 6.0* verwendet. Zusätzlich ist es von Nöten, das Core SDK und das Internet Development SDK aus dem *Microsoft Platform SDK* [Psdk03] zu installieren, in denen sämtliche Interfaces der MMC enthalten sind.

Um den Source Code eines MMC Projekts kompilieren zu können, muss im Visual Studio unter dem Menüpunkt „Extras“ – „Optionen“ der Pfad zum Platform SDK angegeben werden. Die jeweiligen „Inc“- bzw. „Lib“-Verzeichnisse werden bei den „Include-Dateien“ bzw. den „Bibliothekdateien“ im Fenster „Verzeichnisse“ eingetragen und müssen als erster Eintrag in der Liste aufscheinen. Des Weiteren muss man bei den Projekteinstellungen im Link Bereich auf die Bibliotheksdateien „mmc.lib“ und „comctl32.lib“ verweisen. Die MFC Unterstützung sollte nicht verwendet werden.

Wie sich aus den Erfahrungen des ***SAT*** Teams im Umgang mit der MMC Programmierung herausstellte, empfiehlt es sich, den „ATL-COM-Anwendungs-Assistent“ zu verwenden, obwohl vonseiten der Firma *Microsoft* davon abgeraten wird. Diese im Visual Studio integrierte Funktion, die über „Datei“ - „Neu…“ – „Projekte“ ausgeführt werden kann, generiert das Grundgerüst eines neuen Serverprojekts, das wahlweise als in-process Server (.dll Datei) oder als out-of-process Server (.exe Datei) kompiliert wird. Um ein Snap-In Projekt zu realisieren, sollte man die in-process Server Variante verwenden.

Als nächsten Schritt wird dem so erstellten Servergrundgerüst die eigentliche Funktionalität gegeben. Auch hierfür bietet Visual Studio Hilfestellung. Der „ATL Object Wizard“, der unter dem Menüpunkt „Einfügen“ - „Neues ATL-Objekt…“ zu finden ist, bietet die Möglichkeit, unter anderem auch ein MMC Snap-In Objekt dem aktuellen Projekt hinzuzufügen. Als Ergebnis dieser Prozedur werden weitere Dateien generiert, die folgende MMC Basis-Interfaces implementieren:

- **IComponent:**
  … repräsentiert den Result Pane und enthält zu Beginn lediglich eine Routine, die den Nachrichtenverkehr an den im Konsolenbaum übergeordnete Scope Pane Knoten weiterleitet.

- **IComponentData:**
  … stellt den Scope Pane und auch den Wurzelknoten des Snap-Ins dar. Dieses Interface enthält zu Beginn die Grundstruktur zur Initialisierung des Snap-Ins und der damit assoziierten Ressourcen (Symbole, String Tabellen, etc.).

- **CSnapInItemImpl:**
  Diese Klasse ist eigentlich ein Template der ATL und beinhaltet die Basisfunktionen wie Nachrichtenverarbeitung oder Datenrepräsentation eines Knotens im Snap-In Konsolenbaum, die je nach Bedarf überschrieben werden können.

- **ISnapInAbout:**
  … ist für die Darstellung der Auskunft zum Snap-In (Versionsnummer, Herstellerin-

formationen, Copyright, etc.) verantwortlich, die in der MMC im „Snap-In hinzufügen“ Dialog durch Betätigen des Info-Buttons angezeigt wird.

- **IExtendContextMenu (optional):**
  Durch dieses Interface kann man das Standardkontextmenü eines Knoten um beliebige Einträge und damit verbundene Funktionen erweitern.

- **IExtendPropertySheet (optional):**
  … erweitert einen Knoten um so genannte Eigenschaftsseiten, die zur Benutzerinteraktion, d.h. zur Dateneingabe bzw. zur Konfiguration eines Knotens verwendet werden.

- **IExtendControlBar (optional):**
  Dieses Interface ermöglicht die knotenbezogene Darstellung von Funktionssymbolen in der MMC Kontrollzeile.

- **Persistenz (optional):**
  Persistenz bedeutet in der MMC die Speicherung der Konsolenbaumstruktur und der damit verbundenen Daten. Um Persistenz zu implementieren, gibt es drei verschiedene Interfaces, die in ihrer Funktionsweise leicht differieren.

Die vom Object Wizard generierten Dateien enthalten jedoch nur die grundlegende Infrastruktur des Snap-Ins. Nun gilt es, die gewünschte Funktionalität hinzuzufügen. Hierzu ist es empfehlenswert, für jeden Knotentyp, der über eigenständige Daten und Funktionen verfügen soll, ein eigenes Objekt zu erstellen. Da das hinzugefügte Snap-In Objekt jedoch lediglich einen Knoten beinhaltet (die Klasse, die *CSnapInItemImpl* implementiert), sollte man die generierte Dateien- bzw. Klassenstruktur etwas abändern. Im Falle der SAT2 Viewer Komponente wurde aus der *CSnapInItemImpl*-Klasse ein generisches Template erstellt (siehe Listing 4.2.1), das als Basis für jeden Knotentyp fungiert und deren Funktionen je nach Bedarf vom jeweiligen Knoten überschrieben werden.

```
template <class T> class ATL_NO_VTABLE CSAT2ViewerContainer : public CSnap-
InItemImpl<T> {
public:
```

```
    IConsole* m_pConsoleRes;
    CSAT2ViewerICont();
    virtual ~CSAT2ViewerICont();
    STDMETHOD(QueryPagesFor)(DATA_OBJECT_TYPES type);
    STDMETHOD(GetScopePaneInfo)(SCOPEDATAITEM *pScopeDataItem);
    STDMETHOD(GetResultPaneInfo)(RESULTDATAITEM *pResultDataItem);
    STDMETHOD(Notify)(MMC_NOTIFY_TYPE event, long arg, long param, IComponentData*
      pComponentData, IComponent* pComponent, DATA_OBJECT_TYPES type);
    HRESULT OnAddImages(IImageList *pImageList);
    virtual HRESULT OnShowContextHelp(IDisplayHelp *pDisplayHelp);
    virtual HRESULT OnExpand(BOOL bExpand, IConsole *pConsole, HSCOPEITEM parent);
    virtual HRESULT OnShow(IConsole *pConsole, BOOL bShow);
    virtual HRESULT OnSelect(IConsole *spConsole, IComponent* pComponent );
    virtual HRESULT OnRefresh(IConsole *spConsole);
    virtual HRESULT OnDblClick(IConsole *spConsole);
    virtual HRESULT OnRename(IConsole* pConsole, LPOLESTR param);
    HRESULT OnPropertyChange(IConsole *pConsole);
    virtual HRESULT OnDelete(IConsole *pConsole);
};
```

Listing 4.2.1.: Interface des Snap-In Basistemplate

Um ein Snap-In unter dem weiter oben erwähnten Schlüssel in der Windows Registry registrieren zu können, muss jeder Knoten über einen eindeutigen GUID (**G**lobal **U**nique **ID**entifier) Wert verfügen. Der GUID Wert ist ein COM-spezifischer Schlüsselwert und wird bei der Projekterstellung für den Wurzelknoten automatisch generiert. Anhand dieses GUID Wertes können einzelne Knoten im Namespace eindeutig identifiziert, und so eventuelle Namenskonflikte vermieden werden. Der GUID muss selbständig erstellt werden, für jeden eigenständigen Knotentyp eindeutig sein und in gewissen Darstellungsformaten, den so genannten Clipboard Formats, abgebildet werden:

- ***CMeinKnoten*GUID_NODETYPE**
  Eine weltweit eindeutige, alphanumerische Zeichenkette, die mittels speziell dafür vorgesehener Programme generiert werden muss und im CLSID (**CL**a**S**s **ID**entifier) Format angegeben werden muss, das man, wie folgendes Beispiel zeigt, anschreibt: *{ 0x2729e7b3, 0x9162, 0x436a, { 0x9c, 0xf0, 0x5, 0x34, 0xf7, 0xca, 0x76, 0xde } };*

- **m_NODETYPE**
  Dieser Wert stellt lediglich einen Zeiger auf die Speicheradresse des *GUID_NODETYPE* Wertes dar.

- **m_SZNODETYPE**
  Ein Zeigerwert, der auf die Stringrepräsentation des *GUID_NODETYPE* Wertes verweist. Dieser String muss explizit angelegt werden und wird nicht im CLSID Format, sondern wie folgt dargestellt: "2729E7B3-9162-436A-9CF0-0534F7CA76DE"

- **m_SZDISPLAYNAME**
  Dieser Zeigerwert verweist auf den Namen, mit dem der Knoten im Scope Pane des Snap-In beschriftet wird, und muss ebenfalls als String angelegt werden.

- **m_SNAPIN_CLASSID**
  Ein Zeiger, der auf die Speicheradresse des CLSID des Snap-Ins verweist. Dieser automatisch generierte CLSID hat im Allgemeinen die Bezeichnung: CLSID_*MeinSnapIn* und muss explizit referenziert werden.

Um einen GUID Wert für einen Knotentyp zu generieren, kann man sich beispielsweise mit dem im Visual Studio Paket mitgelieferten (und im Unterverzeichnis „.\common\tools“ befindlichen) Werkzeug *guidgen.exe* behelfen, das verschiedene Darstellungen eines weltweit eindeutigen GUIDs erzeugen kann.

Als nächsten Schritt sollte das Registry Script - die *MeinWurzelknoten*.rgs Datei - angepasst werden. Dieses Script ist für die Eintragung der nötigen Registry Werte zuständig und wird immer ausgeführt, sobald das Service - in diesem Fall das Snap-In - registriert wird. Die Anpassung muss für jeden neuen Knotentyp durchgeführt werden und umfasst 2 Einträge:

- unter ***HKLM - NoRemove Software - NoRemove Microsoft - NoRemove MMC - NoRemove Snapins - ForceRemove {… - Node Types*** muss ***{GUID}*** eingetragen werden

* unter ***HKLM - NoRemove Software - NoRemove Microsoft - NoRemove MMC - NoRemove NodeTypes*** muss die Zeile ***ForceRemove {GUID} { }*** eingetragen werden

Anmerkung: {GUID} bezeichnet hier die Stringrepräsentation des GUID des jeweiligen Knotentyps in geschwungenen Klammern. Um nähere Details zu COM in-process Servern und deren Generierung unter zu Hilfenahme von ATL zu erfahren, sei an dieser Stelle auf [Gri99/S.63 ff] verwiesen. Unter anderem wird dort auch auf die Syntax und Funktionalität der Registry Scripts genauer eingegangen.

Eine weitere Anpassung, die zu Beginn der Implementierung durchgeführt werden sollte, betrifft das *ISnapInAbout*-Interface. Wie schon weiter oben beschrieben ist dieses Interface für die Darstellung der Snap-In Informationen wie beispielsweise Versionsnummer oder Copyright zuständig (siehe Abbildung 4.2.1).

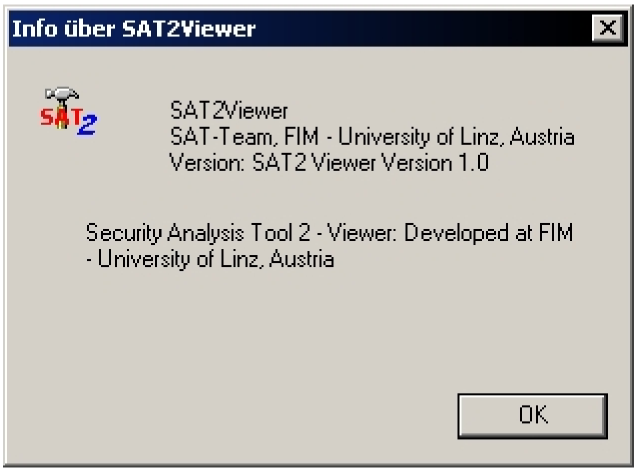


Abbildung 4.2.1: About Information des SAT2 Viewers

Betrachtet man die Standardimplementierung des *ISnapInAbout*-Interfaces, wie sie vom Object Wizard erstellt wird, so bemerkt man, dass 3 Methoden mit Verweisen auf Stringkonstanten und 2 Methoden mit *NULL* Referenzen angeführt sind. Die Verweise auf die Stringkonstanten (in der Form: IDS_*MeinProjekt*_DESC) stellen die möglichen Textinformationen dar, die ohne Schwierigkeiten in der RessourceView – Ansicht unter dem Punkt Stringtabelle editiert werden können. Die Methoden mit den *NULL* Referenzen sind zuständig für die Darstellung des Wurzelknotensymbols (*GetStaticFolderImage*) bzw. des Infobildes (*GetSnapinImage*). Diese beiden Methoden sollten durch den in Listing 4.2.2 angeführten Code ersetzt werden, der die Bildressourcen IDB_*MeinProjekt*_ICO und

IDB_*MeinProjekt*_BMP lädt und die entsprechende Referenz an die jeweiligen Parameter übergibt.

```
STDMETHOD(GetSnapinImage)(HICON *hAppIcon){
	if(hAppIcon == NULL)
		return E_POINTER;
	*hAppIcon = static_cast<HICON>(::LoadImage(_Module.GetResourceInstance(),
		MAKEINTRESOURCE(IDB_MeinProjekt_ICO), IMAGE_ICON, 0, 0,
		LR_DEFAULTCOLOR));
	return S_OK;
}
STDMETHOD(GetStaticFolderImage)(HBITMAP *hSmallImage, HBITMAP *hSmallImageOpen,
      HBITMAP *hLargeImage, COLORREF *cMask){
	*hSmallImage = LoadBitmap(_Module.GetResourceInstance(),
	    MAKEINTRESOURCE(IDB_MeinProjekt_ICO));
	*hSmallImageOpen = LoadBitmap(_Module.GetResourceInstance(),
	    MAKEINTRESOURCE(IDB_MeinProjekt_ICO));
	*hLargeImage = LoadBitmap(_Module.GetResourceInstance(),
	    MAKEINTRESOURCE(IDB_MeinProjekt_BMP));
	return S_OK;
}
```

Listing 4.2.2: Grafikressourcen laden im ISnapInAbout Interface

4.3 MMC Notifications

Die Microsoft Management Console bietet ein grafisches Benutzerinterface und ist somit auf die Interaktion mit dem Benutzer angewiesen. Dabei werden MMC Notifications - im Weiteren als Benachrichtigungen bezeichnet - verwendet, die durch Ereignisse generiert werden, die ihrerseits wiederum durch eine Benutzereingabe ausgelöst werden. Die MMC schickt diese Benachrichtigungen an die *Notify* Methode des *IComponentData*- oder des *IComponent* Interfaces, wobei die Ereignisse entweder mit dem Scope Pane oder mit dem Result Pane assoziiert sind. Im Falle eines Scope Pane Ereignisses wird *IComponentData::Notify* ausgeführt, für den Result Pane *IComponent::Notify*.

Die folgende Tabelle gibt einen Überblick der wichtigsten MMC Benachrichtigungen, auf die ein Snap-In reagieren kann und sollte. Für eine vollständige Liste der Benachrichtigungen sei an dieser Stelle auf [MsLN04] verwiesen.

Benachrichtigung	Beschreibung
MMCN_ADD_IMAGES	Das Snap-In wird aufgefordert eine Bildressource für den Result Pane in den Speicher laden.
MMCN_BTN_CLICK	Ein Symbol in der Symbolleiste der MMC wurde geklickt.
MMCN_CANPASTE_OUTOFPROC	Anfrage der MMC, um festzustellen, ob das Snap-In Einfüge-Operationen aus einem anderen MMC Prozess unterstützt.
MMCN_COLUMN_CLICK	Der Benutzer hat auf die Kopfzeile einer Spalte geklickt (Listendarstellung).
MMCN_COLUMNS_CHANGED	Der Benutzer entfernt Spalten oder fügt sie hinzu in der Listendarstellung.
MMCN_CONTEXTHELP	Die kontextsensitive Hilfe wurde für ein Objekt angefordert.
MMCN_CUTORMOVE	Objekte des Snap-Ins wurden entfernt oder verschoben.
MMCN_DBLCLICK	Ein Doppelklick auf ein Objekt im Scope Pane oder im Result Pane (Listendarstellung).
MMCN_DELETE	Ein Objekt sollte gelöscht werden.
MMCN_EXPAND	Ein Objekt im Scope Pane wird erweitert oder reduziert.
MMCN_EXPANDSYNC	Ein Objekt im Scope Pane wird synchron zum Baumaufbau erweitert.
MMCN_INITOCX	Wird bei der ersten Initialisierung der ActiveXControl-Darstellung gesendet.
MMCN_MINIMIZED	Wird gesendet, sobald ein Fenster des Snap-Ins minimiert oder maximiert wird.
MMCN_PASTE	Benachrichtigt ein Scope Objekt, die ausgewählten Result Objekte einzufügen.
MMCN_PRINT	Ein Benutzer hat eine Druck-Anweisung gegeben.
MMCN_PROPERTY_CHANGE	Informiert das Snap-In über Änderungen der angezeigten Daten.
MMCN_REFRESH	Wird gesendet, sobald Refresh im Kontextmenü selektiert wird.
MMCN_REMOVE_CHILDREN	Alle Kinderknoten ab einem gewissen Knoten sollen gelöscht werden.
MMCN_RENAME	Ein Scope oder Result Objekt wird umbenannt.
MMCN_SELECT	Ein Scope oder Result Objekt wurde selektiert.
MMCN_SHOW	Wird gesendet, immer wenn ein Scope Objekt den Selekt-Fokus erhält oder diesen wieder verliert.
MMCN_SNAPINHELP	Der Benutzer hat eine Help-Anfrage an das Snap-In gesendet.
MMCN_VIEW_CHANGE	Informiert das Snap-In, dass die Darstellung der Objekte aktualisiert werden soll.

Tabelle 4.3.1.: MMC Notifications

In Abhängigkeit von der Art der Benachrichtigung wird von der MMC ein Pointer auf das Data Object des gegenwärtig selektierten Knotens im Scope oder im Result Pane an die *Notify* Methode mitgeschickt. Über diesen Pointer können die nötigen Kontextinformationen abgefragt werden, die es dem Snap-In ermöglichen, objektbezogen auf das Ereignis, dass die Benachrichtigung ausgelöst hat, zu reagieren. In diesen Kontextinformationen ist auch ein Cookie Wert inkludiert, durch den der betroffene Knoten eindeutig identifiziert werden kann.

Wie bereits aus den vorgehenden Ausführungen dieses Kapitels ersichtlich wird, ist die *Notify* Methode für die Koordination der Ereignisverarbeitung eines Snap-Ins verantwortlich. Um die Ereignisse mit größtmöglicher Flexibilität abarbeiten zu können, ist es zweckdienlich, eine

Methode zu implementieren (siehe Listing 4.3.1), die beide *Notify* Methoden (*IComponentData::Notify* und *IComponent::Notify*) ersetzt und die relevanten Benachrichtigungen an eine virtuelle Behandlungsroutine weiterleitet. Diese virtuellen Routinen können dann von den Knotenobjekten, für die die jeweilige Benachrichtigung relevant ist, überschrieben werden. An dieser Stelle sei angemerkt, dass die MMC Standardbehandlungsroutinen für Benachrichtigungen bereitstellt, die zum Einsatz kommen, wenn die *Notify* Methode des Snap-Ins die Konstante *S_FALSE* zurückgibt. Somit empfiehlt es sich für nicht explizit behandelte Nachrichten, einen Defaultwert zu integrieren.

```
STDMETHOD(Notify)( MMC_NOTIFY_TYPE event, long arg, long param, IComponentData*
        pComponentData, IComponent* pComponent, DATA_OBJECT_TYPES type){
        HRESULT hr = S_FALSE;
        CComPtr<IConsole> spConsole;
        CComQIPtr<IHeaderCtrl2, &IID_IHeaderCtrl2> spHeader;
        _ASSERTE(pComponentData != NULL || pComponent != NULL);
        if(pComponentData != NULL)
                spConsole =((CSAT2Viewer*)pComponentData)->m_spConsole;
        else {
                spConsole =((CSAT2ViewerComponent*)pComponent)->m_spConsole;
                spHeader = spConsole;
                // save pointer to Console for later Refresh
                if(!m_pConsoleRes)
                                m_pConsoleRes = spConsole;
        }
        switch(event){
                case MMCN_SHOW: {
                        hr=OnShow(spConsole, arg);
                        break;
                }
                case MMCN_EXPAND: {
                        hr = OnExpand(true, spConsole, param);
                        break;
                }
                case MMCN_ADD_IMAGES:{
                        IImageList* pImageList =(IImageList*) arg;
                        hr = OnAddImages(pImageList);
                        break;
                }
                case MMCN_DBLCLICK:{
                        hr = OnDblClick(spConsole);
                        break;
                }
                case MMCN_SELECT:{
```

```
                hr = OnSelect(spConsole, pComponent);
                break;
            }
            case MMCN_REFRESH:{
                hr = OnRefresh(spConsole);
                break;
            }
            case MMCN_RENAME:   {
                hr = OnRename(spConsole,(LPOLESTR)param);
                break;
            }
            case MMCN_PROPERTY_CHANGE:{
                hr = OnPropertyChange(spConsole);
                break;
            }
            case MMCN_DELETE:{
                hr = OnDelete(spConsole);
                break;
            }
            case MMCN_COLUMNS_CHANGED:{
                hr = S_OK;
                break;
            }
        }
        return hr;
}
```

Listing 4.3.1: Notify Methode des Snap-In Basistemplate

4.4 Der Scope Pane

Der Scope Pane ist die Ausgangsbasis einer jeden Snap-In Anwendung. Beginnend beim Wurzelknoten kann je nach Bedarf der Anwendung eine Baumstruktur im Scope Pane aufgebaut werden. Um dieses Baumkonzept und die damit verbundene Nomenklatur zu verstehen, sollte man folgende Definition bedenken, die aus [Sed92/S.58 ff] übernommen wurde:

„Ein Baum ist eine nichtleere Menge von Knoten[…]. Ein Knoten ist ein einfaches Objekt, das einen Namen haben und andere mit ihm verknüpfte Informationen tragen kann. […] Einer der Knoten im Baum wird als die Wurzel bezeichnet. […] Jeder Knoten (außer der Wurzel) besitzt genau einen Knoten, der sich unmittelbar über ihm befindet und als sein direkter

Vorgänger (parent) bezeichnet wird; die Knoten unmittelbar unter einem Knoten werden seine direkten Nachfolger (children) genannt. […] Knoten ohne Nachfolger werden manchmal Blätter oder Endknoten genannt.“

Der hierarchische Aufbau des Baumkonzepts in einer Snap-In Anwendung impliziert eine Vater–Sohn–Abhängigkeit, die sich im Scope Pane widerspiegelt. Zur Umsetzung dieses Konzepts ist es nötig, eine Datenstruktur in jedem direkten Vorgänger- oder Vaterknoten zu implementieren, welche die darunter liegenden Söhne referenziert. Im Regelfall ist eine einfach verkettete Liste von Zeigern auf die jeweiligen Knoten, die eingehängt werden, ausreichend um eine solche Struktur aufbauen zu können. Bedient man sich beispielsweise der *list* Klasse aus der Standard Template Library, so ermöglicht dies eine Abarbeitung der direkten Nachfolgerknoten mittels eines Iterators, was sich in der Snap-In Programmierung als praktisch erweist.

Jeder Knoten muss selbständig dafür sorgen, dass seine direkten Nachfolger im Speicher angelegt und auch wieder entfernt werden. Dies sollte, sofern es nicht dem gewünschten Programmablauf widerspricht, im Konstruktor bzw. im Destruktor des (Vater-)Knotenobjekts abgehandelt werden. Der eigentliche Baumaufbau findet durch Abarbeitung der *Expand* Nachricht (siehe Kapitel 4.3) statt, die von der MMC an die *IComponentData* Implementierung des Knotens gesendet wird, sobald dieser selektiert wird. Um einen zusätzlichen Knoten im Scope Pane an einen bereits geöffneten Pfad anzuhängen, sollte man bedenken, dass danach eine *Refresh* Nachricht an die Management Console geschickt werden muss, die eine Neuzeichnung des Scope Pane Baumes initiiert.

Die Darstellung der Knoten im Scope Pane basiert auf der ScopeDataItem (SDI) Datenstruktur, die jeder Knoten implementieren muss (siehe Listing 4.4.1) und aus folgenden Komponenten aufgebaut ist:

- ***DWORD mask:***
  Ein 32 Bit Zahlenwert (unsigned), der ein Feld von Flags beschreibt. Dieses Feld fungiert als Schlüssel der SDI Datenstruktur und zeigt an, welche der nachfolgenden Datenstrukturkomponenten einen gültigen Wert enthält. Die zu setzende Konstante wird nachfolgend in Klammer bei jeder Komponente mit angeführt.

- ***LPOLESTR displayname:***

  Diese Komponente kann entweder einen Zeiger auf einen *NULL* terminierten String, der den Namen des Knoten beschreibt, oder die Konstante *MMC_CALLBACK* enthalten. Da einige Methoden der MMC jedoch nur korrekt funktionieren, wenn *MMC_CALLBACK* gesetzt ist, empfiehlt es sich, den Bildschirmnamen eines Knoten in der Variable *m_SZDISPLAYNAME* (siehe Kapitel 4.2) zu speichern. (`mask = mask || SDI_STR`)

- ***SHORT nImage:***

  Ein 16 Bit Zahlenwert, der die Position des für den Knoten zu verwendenden Symbols in der Symbolressource (siehe später in diesem Kapitel) bezeichnet. `(mask = mask || SDI_IMAGE)`

- ***int nOpenImage:***

  Ähnlich wie *nImage* bezieht sich dieser Wert auf die Symbolressource. Dabei wird jedoch das Symbol indiziert, das die Management Console anzeigt, wenn der Knoten selektiert bzw. geöffnet wird.
  `(mask = mask || SDI_OPENIMAGE)`

- ***UINT nState:***

  Dieser Wert ist nur für die Funktion *IConsoleNameSpace2::GetItem* relevant, die anhand dieser Bitmaske zurückgibt, ob der Knoten schon erweitert bzw. geöffnet wurde oder nicht.
  `(mask = mask || SDI_STATE)`

- ***int cChildren:***

  Ein Zahlenwert, der die Anzahl der Kinderknoten enthält. Wenn ein Knoten zum Zeitpunkt des Einhängens in den MMC Baum absehbar keine direkten Nachfolger hat, sollte dieser Wert auf 0 gesetzt werden. Ansonsten sollte die Anzahl der bereits existierenden Nachfolgeknoten hier gesetzt werden und etwaige später dazukommende Knoten durch *IConsoleNameSpace2::SetItem* ergänzt werden. `(mask = mask || SDI_CHILDREN)`

- ***LPARAM lParam:***

  Ein Zeiger auf einen 32 Bit Wert, der mit dem Knoten assoziiert wird und als Cookie dient. Dieser Wert sollte ein Zeiger auf den Knoten selbst sein, der dementsprechend gecastet werden muss.

  `(mask = mask || SDI_PARAM)`

- ***HSCOPEITEM ID:***

  Dieser Wert spezifiziert einen eindeutigen Bezeichner für den Knoten. Nachdem ein Knoten mit der Methode *IConsoleNameSpace2::InsertItem* in den Namespace eingefügt wurde, sorgt die Management Console dafür, dass eine eindeutige ID in dieser Datenkomponente abgelegt wird, anhand derer der Knoten im Namespace identifiziert wird.

- ***HSCOPEITEM relativeID:***

  Die Knoten der MMC werden relativ zu den anderen Knoten in die Baumstruktur eingefügt. Dieser Wert enthält folglich die ID eines relativen Vorgängers, wobei der Bezug zum Vorgänger durch ein Flag in *mask* interpretiert wird:

  - *SDI_PARENT:* … bedeutet, dass *relativeID* die *HSCOPEITEM ID* des Vaters ist und dass der Knoten als letzter Sohn eingefügt wird.

  - *SDI_PREVIOUS:* … *relativeID* ist die ID des Vorgängers in derselben Baumebene

  - *SDI_NEXT:* … *relativeID* ist die ID des Nachfolgers in derselben Baumebene

  - *SDI_FIRST:* … wie *SDI_PARENT*, jedoch wird der Knoten als erster Sohn eingehängt

```
memset(&m_scopeDataItem, 0, sizeof(SCOPEDATAITEM));
m_scopeDataItem.mask = SDI_STR | SDI_IMAGE | SDI_OPENIMAGE |
                       SDI_PARAM | SDI_CHILDREN;
m_scopeDataItem.displayname = MMC_CALLBACK;
m_scopeDataItem.nImage = 2;
m_scopeDataItem.nOpenImage = 3;
```

```
m_scopeDataItem.lParam =(LPARAM) this;
m_scopeDataItem.cChildren = 3;
```

Listing 4.4.1: Beispiel für SDI Datenstruktur

Die Visualisierung der Knoten passiert in der MMC mittels Symbolen, wobei sich jeder Knoten in zwei Zuständen befinden kann. Der geschlossene Zustand ist der Ausgangszustand. Dieser geht in den geöffneten Zustand über, sobald der Knoten mittels des dafür vorgesehenen Pluszeichens erweitert wird. Steigt man nun in der Baumhierarchie tiefer, so bleibt der Knoten geöffnet. Dieser Zustand wird erst dann verlassen, wenn der Knoten wieder reduziert wird. Beiden Zuständen kann man über die SDI Struktur (siehe weiter oben) ein individuelles Symbol aus einer Symbolressourcendatei zuordnen, die in das Visual C++ Projekt eingebunden werden muss. Dabei sollte man darauf achten, dass die beiden vom Object Wizard erstellten Symbolressourcen – wobei eine Ressource 16x16 Pixel große Symbole und eine zweite 32x32 Pixel große Symbole enthält – mit Symbolen im richtigen Format erweitert werden. Im Visual Studio 6 lässt sich dies ohne größere Mühe bewerkstelligen. Wichtig ist, dass die Symbolressourcen immer 16 bzw. 32 Pixel hoch sind und ein Vielfaches von 16 bzw. 32 Pixel breit. Ansonsten verweigert die Management Console den Zugriff auf diese Ressourcen und benutzt stattdessen die Standardsymbole.

4.5 Der Result Pane

Wird ein Knoten im Scope Pane selektiert, so erscheint die Repräsentation der mit dem Knoten assoziierten Daten im Result Pane. Diese Repräsentation kann auf verschiedene Weise dargestellt werden (siehe Kapitel 4.1).

Ähnlich der Datenrepräsentation eines Scope Knotens liegt auch einem Knoten im Result Pane eine eigene Datenstruktur zugrunde. Diese ResultDataItem (RDI) Datenstruktur, die ebenfalls von jedem Knoten implementiert werden muss (siehe Listing 4.5.1), besteht aus folgenden Attributen:

- ***DWORD mask:***
  Ein 32 Bit Wert, der wie bei der SDI Datenstruktur (siehe Kapitel 4.4) ein Feld von Konfigurationsflags repräsentiert. Dieser Wert bezieht sich auf die RDI Struktur und beschreibt, welche der nachfolgenden Attribute einen gültigen Wert enthält. Die zu

setzende Konstante wird nachfolgend in Klammer bei jeder Komponente mit angeführt.

- ***BOOL bScopeItem:***
  Ein boolescher Wert, der sich auf das weiter unten beschriebene Attribut *lParam* bezieht. Dieser Wert wird auf wahr gesetzt, wenn sich *lParam* auf einen Scope Knoten bezieht. Bezieht sich *lParam* auf einen Result Knoten, so wird der Wert auf falsch gesetzt.

- ***HRESULTITEM itemID:***
  Ein eindeutiger Wert, der von der Management Console gesetzt wird, sobald der Result Knoten in den Baum eingefügt wird, und der dazu dient, den Knoten im Speicherbereich des Snap-In zu identifizieren.

- ***int nIndex:***
  Dieser Wert spezifiziert den (0-basierten) Index des Knotens in einer dem Knoten übergeordneten Datenstruktur, die vom Knoten referenziert wird.
  `(mask = mask || RDI_INDEX)`

- ***int nCol:***
  Dieser Wert gibt den Index der Spalte an, auf die eine Operation ausgeführt werden soll. Ist dieser Wert 0, so beziehen sich sämtliche Operationen auf das Knotenobjekt und nicht auf eine einzelne Spalte.

- ***LPOLESTR str:***
  Dieses Attribut kann entweder einen Zeiger auf einen *NULL* terminierten String, der den Namen des Knoten im Result Pane darstellt, oder die Konstante *MMC_CALLBACK* enthalten.
  `(mask = mask || RDI_STR)`

- ***SHORT nImage:***
  Ein 16 Bit Zahlenwert, der die Position des zu verwendenden Symbols in der Symbol-

ressource bezeichnet, dass für den Result Knoten verwendet werden soll. `(mask = mask || RDI_IMAGE)`

- ***UINT nState:***
  Dieses Attribut spezifiziert eine Zustandsbitmaske für den Result Knoten und kann nachfolgende Konstanten beinhalten:

  - *LVIS_CUT:* … bedeutet, dass dieser Knoten für eine Ausschneide- und Einfügeoperation markiert ist.

  - *LVIS_DROPHILITED:* ... bezeichnet den Zustand, in dem sich der Knoten befindet, sobald er in eine Drag-and-Drop Operation involviert ist.

  - *LVIS_FOCUSED:* … bedeutet, dass dieser Knoten gegenwärtig den Fokus im Result Pane besitzt.

  - *LVIS_SELECTED:* … zeigt an, dass der jeweilige Knoten selektiert wurde. Im Result Pane können mehrere Knoten gleichzeitig selektiert werden, aber nur jeweils ein Knoten erhält den Fokus.

  `(mask = mask || RDI_STATE)`

- ***LPARAM lParam:***
  Ein Zeiger auf einen 32 Bit Wert, der mit dem Knoten assoziiert wird und als Cookie dient. Dieser Wert kann wahlweise auf das Cookie des übergeordneten Scope Knotens oder auf einen gecasteten Cookie-Wert des Result Knoten selbst zeigen (siehe *bScopeItem*).
  `(mask = mask || SDI_PARAM)`

- ***int iIndent:***
  Ein intern reservierter Wert, der nicht berücksichtigt werden muss. `(mask = mask || RDI_INDENT)`

```
      memset(&m_resultDataItem, 0, sizeof(RESULTDATAITEM));
      m_resultDataItem.mask = RDI_STR | RDI_IMAGE | RDI_PARAM;
      m_resultDataItem.str = MMC_CALLBACK;
      m_resultDataItem.nImage = INDEX_GROUP_BMP;
      m_resultDataItem.lParam =(LPARAM) this;
      m_resultDataItem.nCol = 0;
```

Listing 4.5.1: Beispiel für RDI Datenstruktur

Die Darstellung, die als gebräuchlichste Variante in den üblichen MMC Anwendungen (siehe Tabelle 4.1.1) wie auch im ***SAT*** vorherrscht, ist die Listendarstellung. Der Name Listendarstellung rührt verständlicherweise daher, dass alle Söhne des selektierten Knotens und die damit verbundenen Daten – im Standardfall ist dies lediglich der Name des Subknotens – im Result Pane aufgelistet werden. Um jedoch einen höheren Informationsgehalt im Result Pane zu visualisieren, bedarf es einer Überarbeitung der Methode zur Behandlung der *MMCN_SHOW* Benachrichtigung (siehe Kapitel 4.3). Diese Methode, die in der *Notify* Routine des ***SAT*** Basistemplate (siehe Listing 4.3.1) mit *OnShow* adressiert wird, ist von jedem Knoten, der über direkte Nachfolgerknoten verfügt, zu implementieren (siehe Listing 4.5.2).

```
HRESULT CMemberFolder::OnShow(IConsole *pConsole, BOOL bShow) {
  HRESULT hResult = S_OK;
  IHeaderCtrl2 *pHeaderCtrl = NULL;
  IResultData *pResultData = NULL;
  if(bShow) {
      CComQIPtr<IResultData,&IID_IResultData> spResultData(pConsole);
      CComQIPtr<IHeaderCtrl2,&IID_IHeaderCtrl2> spHeaderCtrl(pConsole);
      hResult = spHeaderCtrl->InsertColumn(0, L"S. Principal ", 0, 120);
      hResult = spHeaderCtrl->InsertColumn(1, L"Name ", 0, 150);
      hResult = spHeaderCtrl->InsertColumn(2, L"Full Name ", 0, 150);
      hResult = spHeaderCtrl->InsertColumn(3, L"Comment ", 0, MCLV_AUTO);
     hResult = spHeaderCtrl->InsertColumn(4, L"Location ",0, HIDE_COLUMN);
      // Set view style
      hResult = spResultData->ModifyViewStyle((MMC_RESULT_VIEW_STYLE)NULL,
      MMC_NOSORTHEADER);
      hResult = spResultData->DeleteAllRsltItems();
      std::list<CMemberItem*>::iterator li;
      for(li = m_Children.begin(); li != m_Children.end(); li++)
            hResult = spResultData->InsertItem(&(*li)->m_resultDataItem);
      hResult = spResultData->Sort(0, 0, 0);
      hResult = spResultData->Sort(1, 0, 0);
  }
```

```
  return hResult;
}
```

Listing 4.5.2: Beispiel einer OnShow Implementierung

Die Manipulation der Standardanzeige wird durch die beiden Interfaces *IResultData* und *IHeaderCtrl2* ermöglicht. *IHeaderCtrl2* implementiert Methoden, mit deren Hilfe die Spalten im Result Pane modifiziert werden. Mittels der Methode *InsertColumn* wird eine neue Spalte angelegt und deren Name, Reihenfolge und Erscheinungsverhalten definiert. *IResultData* wird benötigt, um die direkten Nachfolgerknoten in die Result Pane Anzeige – mittels *InsertItem* - einzufügen und um die Sortierungskriterien festzulegen. Mittels *ModifyViewStyle* wird bestimmt, wie das Snap-In die angezeigten Daten bei einem durch den Benutzer ausgelösten Ereignis weiter verarbeitet.

Für die eigentlichen Daten, die in der erweiterten Anzeige dargestellt werden, sind die direkten Nachfolgerknoten selbst verantwortlich. Um die einzelnen Einträge in den Result Pane Spalten entsprechend ihrer Definition korrekt anzuzeigen, muss ein Snap-In Knoten die Methode *GetResultPaneColInfo* implementieren. Diese Routine muss je nach Spaltenindex, der ihr übergeben wird, die entsprechende Stringrepräsentation der gewünschten Daten liefern. Hierbei ist darauf zu achten, dass der String in einem COM kompatiblen Format vorliegt oder in dieses entsprechend umgewandelt wird.

4.6 Menüführung

Die Interaktion des Benutzers mit dem Snap-In wurde bereits in Kapitel 4.3 besprochen und auf die Nachrichtenverarbeitung zurückgeführt. Dieses Kapitel soll weiterführend erklären, welche Möglichkeiten die MMC noch bietet, um den Bedürfnissen eines Administrators bei der täglichen Arbeit gerecht zu werden und wie diese umzusetzen sind. Zu diesem Zweck wird vorgestellt, wie das vordefinierte Menü eines Snap-In modifiziert und erweitert werden kann, um dem Benutzerkomfort im Umgang mit dem Snap-In zu erhöhen.

Die Microsoft Management Console bietet eine Reihe von Standard Menüoptionen an, die in der Tabelle 4.6.1 aufgeführt sind.

Funktion	MMC Kommandokennung
Ausschneiden	***MMC_VERB_CUT***
Kopieren	***MMC_VERB_COPY***
Einfügen	***MMC_VERB_PASTE***
Löschen	***MMC_VERB_DELETE***
Eigenschaften	***MMC_VERB_PROPERTIES***
Umbenennen	***MMC_VERB_RENAME***
Aktualisieren	***MMC_VERB_REFRESH***
Drucken	***MMC_VERB_PRINT***
Öffnen	***MMC_VERB_OPEN***

Tabelle 4.6.1: Standard Menüoptionen

Um einer dieser Standard Menüeinträge zu aktivieren, muss man die *MMCN_SELECT* Benachrichtigung (Kapitel 4.3) behandeln und entsprechend erweitern. Setzt man die Routine des Basistemplates (Listing 4.3.1) voraus, so genügt ein Aufruf der *IConsoleVerb* Methode *SetVerbState* in *OnSelect* um die Aktivierung durchzuführen. Dabei muss diese Methode als ersten Parameter die Kommandokennung erhalten, die in der obigen Tabelle als Zusatz zum jeweiligen Menüeintrag angeführt ist. Einen Zeiger auf das *IConsoleVerb* Interface liefert folgende Kommandozeile, die sich des der Methode *OnSelect* übergebenen Zeigers auf das *IConsole* Interface bedient:

```
spConsole->QueryConsoleVerb(&pConsoleVerb);
```

Wird ein aktivierter Standard Menüeintrag vom Benutzer ausgewählt, so wird von der MMC eine entsprechende Benachrichtigung an die *Notify* Routine des jeweiligen Knotens gesendet. Ausnahmen dieser Regelung bilden die Einträge Eigenschaften, Kopieren und Ausschneiden. Der Eintrag Eigenschaften veranlasst die MMC *CreatePropertyPages* (siehe Kapitel 4.7) aufzurufen, mit der Folge, dass ein Eigenschaftsdialog angezeigt wird, falls dieser implementiert wurde. Im Falle der Einträge Kopieren und Ausschneiden wird die Methode *QueryDataObject* des jeweiligen *IComponent* bzw. *IComponentData* Interfaces aufgerufen.

An dieser Stelle sei erwähnt, dass die Microsoft Management Console auch Drag & Drop Funktionalität unterstützt. Hierzu müssen die 3 Menüeinträge Ausschneiden, Kopieren und Löschen aktiviert sein und richtig abgehandelt werden. Ansonsten muss kein zusätzlicher Kodierungsaufwand betrieben werden.

Es besteht weiters die Möglichkeit, einen Menüeintrag als Standardeintrag auszuwählen. Dieser Eintrag wird im Kontextmenü in einem fetteren Schriftgrad dargestellt und wird automatisch mit jedem Doppelklick auf den jeweiligen Knoten aufgerufen. Im Regelfall ist dies der Eintrag „Öffnen“. Mittels der Methode *SetDefaultVerb* des *IConsoleVerb* Interfaces, die wiederum die Kommandokennung des gewünschten Menüeintrags als Parameter benötigt, wird der Standardeintrag dementsprechend umgesetzt.

Um einen Knoten des Snap-Ins mit einem eigenen Menüeintrag zu erweitern, muss der gewünschte Eintrag in der vordefinierten Menüressource im Visual Studio Ressourceneditor vorgenommen werden. Dabei sollte man für jeden Knoten, der über ein eigenes Menü verfügen soll, eine eigene Menüressource anlegen.

Nachdem die gewünschte Menüoption eingetragen wurde, muss eine Behandlungsroutine implementiert werden. In dieser Routine wird verständlicherweise die gewollte Funktionalität, die sich hinter der Menüoption verbirgt, abgearbeitet. Die Aufrufsdefinition der Behandlungsroutine, die in der Headerdatei eingetragen werden muss, wird auf folgende Art angeführt:

```
STDMETHOD(OnMeinMenueEintrag)(bool& bHandled, CSnapInObjectRootBase* pObj);
```

Damit das Menü mit dem zugehörigen Knoten assoziiert wird und damit die erstellte Behandlungsroutine auf den entsprechenden Menüeintrag korrekt reagiert, bedient man sich folgendes Kommandomappings:

```
SNAPINMENUID(IDR_MeinMenue)
BEGIN_SNAPINCOMMAND_MAP(CMeinKnoten, FALSE)
      SNAPINCOMMAND_ENTRY(ID_MeinMenueEintrag, OnMeinMenueEintrag)
END_SNAPINCOMMAND_MAP()
```

4.7 Eigenschaftsdialog und Persistenz

Die MMC Dateneingabe wird über so genannte Eigenschaftsdialoge abgewickelt. Ein Eigenschaftsdialog ist ein Fenster, das dem Benutzer die Möglichkeit bietet, die Eigenschaften eines Knotens zu betrachten oder zu ändern. Innerhalb eines solchen Fensters präsentieren sich untergeordnete Eigenschaftsseiten, die nach semantisch zusammenhängenden Einstellungen gruppiert sind und im üblichen Windows Look & Feel, d.h. durch Karteireiter separiert, dargestellt werden.

Die Erweiterung eines Knotens um einen Eigenschaftsdialog bedarf mehrerer Schritte. Zunächst sollte man sicherstellen, dass das *IExtendedPropertySheet2* Interface bzw. dessen ATL Pendant *IExtendPropertySheetImpl* von der Klasse implementiert wird, die für die Kommunikation des Snap-In mit der MMC verantwortlich ist. Falls der Eigenschaftsdialog für einen Knoten im Scope Pane aufgerufen werden soll, so handelt es sich dabei um die Klasse, die *IComponentData* implementiert. Wird der Eigenschaftsdialog für einen Knoten im Result Pane benötigt, so muss man die Klasse mit der *IComponent* Implementierung heranziehen.

Als nächsten Schritt wird der eigentliche Eigenschaftsdialog entwickelt. Dazu leitet man eine Klasse von dem ATL Interface *CSnapInPropertyPageImpl* ab und überschreibt die relevanten Methoden, in erster Linie die Methoden *OnInitDialog* und *OnApply*. Wie die Namen der Methoden schon vermuten lassen, handelt es sich bei *OnInitDialog* um die Initialisierungsroutine des Eigenschaftsdialogs. *OnApply* wird ausgeführt, sobald der Benutzer die Änderungen, die in den Dialogfeldern gemacht wurden, mittels OK bestätigt. In *OnInitDialog* sollte man sicherstellen, dass alle Felder der jeweiligen Eigenschaftsseiten mit den korrekten Daten gefüllt werden. In *OnApply* müssen diese Dialogfelder folglich in die Datenstruktur des aufrufenden Knotens gespeichert werden. Am Ende der *OnApply* Routine muss die MMC über die Änderung in Kenntnis gesetzt werden. Dies geschieht mittels des Aufrufs der Routine *MMCPropertyChangeNotify*, die eine *MMCN_ PROPERTY_CHANGE* Benachrichtigung (siehe Kapitel 4.3) absetzt. Im Konstruktor der Dialogklasse sollte schließlich noch ein Zeiger auf den aufrufenden Knoten gesetzt werden, um den Datenaustausch zwischen dem aufrufenden Knoten und dem Eigenschaftsdialog flexibler gestalten zu können.

Um das Erscheinungsbild festzulegen, bedient man sich des Ressourceneditors, der im Visual Studio integriert ist. Dieser Editor sollte dazu benutzt werden, um die Eingabefelder zu definieren, die logischerweise die Eigenschaften bzw. die Daten eines Knotens widerspiegeln. Weiters lassen sich auch noch Erscheinungs- und Verarbeitungsmerkmale einstellen, die den Umgang mit den Eigenschaftsseiten prägen. Der Name dieser Ressource, die auf diese Weise erstellt wurde, muss letztlich noch in der Headerdatei der Eigenschaftsseite mittels

```
enum { IDD = IDD_Name_der_Eigenschaftsseite };
```

referenziert werden.

Die Instanziierung des Eigenschaftsdialogs passiert in der Routine *CreatePropertyPages* (siehe Listing 4.7.1), die jeder Knoten, für den der entsprechende Dialog aufgerufen werden soll, überschreiben muss. Die übergebene Variable *type* kann dazu verwendet, um zu unterscheiden, ob der Aufruf von der Scope Pane oder der Result Pane Repräsentation des Knotens stammt. Somit hat man die Möglichkeit, zwei verschiedene Eigenschaftsdialoge für einen Knoten anzubieten. Mittels der Methode *Create* wird der Dialog in den MMC Speicherbereich geladen, *AddPage* bewirkt das Anzeigen des Dialogs.

Wie schon zuvor erwähnt ist es weiters möglich, mehrere Eigenschaftsseiten in einem Dialog zusammenzufassen, um eine semantische Gruppierung der Eingaben zu erreichen. Um dies zu bewerkstelligen, bedient man sich der HPROPSHEETPAGE Struktur, die ein Feld von Eigenschaftsseiten bezeichnet. Die Methode *CreatePropertySheetPage* ersetzt in diesem Falle die *Create* Methode.

```
HRESULT CQuery::CreatePropertyPages(LPPROPERTYSHEETCALLBACK lpProvider,
    long handle, IUnknown* pUnk, DATA_OBJECT_TYPES type) {
  if(type == CCT_SCOPE){
       CQVPropPage* pPage = new CQVPropPage(this, handle, true, _T("New Query"));
       HRESULT hr = lpProvider->AddPage(pPage->Create());
       return S_OK;
  } else if(type == CCT_RESULT) {
       CQVPropPageGen* pPPQueryGeneral = new CQVPropPageGen(this, handle, false,
              _T("General"));
       CQVPropPageOpt* pPPQueryOptions = new CQVPropPageOpt(this, handle, false,
              _T("Options"));
       CQVPropPageAdv* pPPQueryAdvanced = new CQVPropPageAdv(this, handle, false,
              _T("Advanced"));
       HPROPSHEETPAGE* hPPages = new HPROPSHEETPAGE[3];
       hPPages[0] = ::CreatePropertySheetPage(&pPPQueryGeneral->m_psp);
       hPPages[1] = ::CreatePropertySheetPage(&pPPQueryOptions->m_psp);
       hPPages[2] = ::CreatePropertySheetPage(&pPPQueryAdvanced->m_psp);
       // Add the pages to the console's Property Sheet
       lpProvider->AddPage(hPPages[0]);
       lpProvider->AddPage(hPPages[1]);
       lpProvider->AddPage(hPPages[2]);
       delete [] hPPages;
       return S_OK;
  }

  return E_UNEXPECTED;
}
```

Listing 4.7.1: CreatePropertyPages Methode für den SAT Query Dialog

Als letzten Schritt bedarf es noch der Behandlung der Frage, wie der eigentliche Aufruf des Dialogs zustande kommt. Wie bereits im Kapitel 4.6 erwähnt, stellt die MMC eine Reihe von Standard Menüoptionen zur Verfügung, die lediglich aktiviert werden müssen. Eine dieser Optionen ist der Menüpunkt Eigenschaften, der genau zu dem Zweck konzipiert wurde, den Eigenschaftsdialog aufzurufen. Damit jeder Knoten bereits ab seinem ersten Erscheinen im Snap-In über diese Standard Menüoption verfügt, muss der Wert *MMC_VERB_PROPERTIES* in der *OnSelect* Routine mittels *SetVerbState* gesetzt werden.

Der Aufruf des Eigenschaften-Menüeintrags durch den Benutzer hat zur Folge, dass die MMC die Routine *QueryPagesFor* des entsprechenden Knotens ausführt. Diese muss *S_OK* zurückgeben, um der MMC zu signalisieren, dass ein Eigenschaftsdialog für den Knoten existiert. In weiterer Folge wird von der MMC die bereits weiter oben beschriebene Methode *CreatePropertyPages* aufgerufen und der Eigenschaftsdialog wird angezeigt.

Da es verständlicherweise für Administrationsanwendungen einen Sinn ergibt, die auf die eben beschriebene Weise gewonnenen Daten persistent im Speicher zu halten, existiert ein Snap-In Daten Persistenz Modell. Dieses Modell erlaubt dem Anwender, den aktuellen Zustand der Applikation, d.h. die Baumstruktur der Knoten inklusive deren knotenbezogene Daten, in einer .MSC Datei beispielsweise auf die Festplatte zu speichern.

Dem Persistenz Modell liegen 3 Interfaces zugrunde: *IPersistStream*, *IPersistStreamInit* und *IPersistStorage*. Um die Snap-In Daten persistent zu halten, genügt es, eines der drei Interfaces zu implementieren. Die Vorgehensweise der MMC um festzustellen, ob das Snap-In Persistenz unterstützt, passiert folgendermaßen: die MMC fragt zuerst das *IPersistStream* Interface ab. Wird hier kein Zeiger auf ein solches Interface zurückgegeben, so wird *IPersistStreamInit*, und, falls dies fehlschlägt, *IPersistStorage* abgefragt. Die Befehlsreferenz des Microsoft Platform SDK empfiehlt, wahlweise das *IPersistStream* oder das *IPersistStreamInit* Interface zu implementieren, da sich diese im Umfang einfacher und in der Umsetzung unkomplizierter erweisen. Im Gegensatz dazu verlangt das *IPersistStorage* Interface die Implementierung von zahlreichen Methoden, die nicht von einem Snap-In benötigt werden um den Datenbestand zu sichern.

Es existieren 5 Methoden, die vom *IPersistStream* bzw. *IPersistStreamInit* Interface vorausgesetzt werden, um Persistenz zu gewährleisten, und sollten in der *IComponentData* Implementierung umgesetzt werden:

- GetClassID
- GetSizeMax
- IsDirty
- Load
- Save

Die wichtigsten dieser Methoden sind die *Load* und die *Save* Methode. *Load* wird immer dann aufgerufen, sobald der Benutzer eine .MSC öffnet. Die *Save* Methode (siehe Listing 4.7.2) kommt zum Einsatz, wenn der Benutzer im MMC Menü „Speichern" bzw. „Speichern unter" ausführen lässt. Für die *Save* Methode ist es wichtig, die *IsDirty* Methode konsistent zu halten.

```
HRESULT CSAT2Viewer::Save(IStream *pStm, BOOL fClearDirty){
  HRESULT hr;
  ULONG dataSize, written;
  QUERY_PROPERTIES_ST* saveQ;
  ULONG dataSize2 = sizeof(int);
  if(fClearDirty)   // reset the dirty bit
       m_pNode->m_pQV->bIsDirty = false;
  std::list<CQuery*>::iterator li;
  Storage* st1 = new Storage();
  st1->iSize =m_pNode->m_pQV->m_Children.size();
  pStm->Write(st1, sizeof(Storage), NULL);
  for(li = m_pNode->m_pQV->m_Children.begin(); li != m_pNode->m_pQV->
       m_Children.end(); li++){
       dataSize = sizeof(SQueryPropertiesStorage);
       saveQ = new QUERY_PROPERTIES_ST();
       strcpy(saveQ->sQueryName,(*li)->mQProps->sQueryName.c_str());
       saveQ->iScanVersion =(*li)->mQProps->iScanVersion;
       // …
       HRESULT hr = pStm->Write(saveQ, dataSize, &written);
       delete saveQ;
  }
  delete st1;
  return hr;
}
```

Listing 4.7.2: Save Methode im IComponentData des SAT

IsDirty soll der MMC signalisieren, dass sich im zu speichernden Datenbestand etwas geändert hat. Hier muss in diesem Falle *S_OK* zurückgegeben werden. Ist der Datenbestand noch nicht geändert worden, sollte *S_FALSE* returniert werden. Um die Konsistenz dieser Methode zu sichern, ist es ratsam, eine boolesche Variable in der *IComponentData* Klasse mitzuführen.

Die *GetSizeMax* Methode erhält im Aufruf einen Ausgangsparameter, der die Größe des zu speichernden Datenstroms in Bit erhalten soll. *GetClassID* wird ebenfalls mit einem Ausgangsparameter aufgerufen. Dieser sollte auf den CLSID Wert des Snap-Ins zeigen. Beide Methoden müssen schließlich *S_OK* zurückliefern, um der MMC zu signalisieren, dass ihre Funktionalität korrekt implementiert wurde.

4.8 Weiterführende Überlegungen

Wie aus diesem kurzen Leitfaden resultiert, stellt die Microsoft Management Console ein Framework dar, das es ermöglicht, COM-basierte Snap-Ins für Administrationsanwendungen zu entwickeln. Dabei wurde die Vielfalt der Möglichkeiten der MMC Programmierung aus aufwands- und platztechnischen Gründen nur oberflächlich angekratzt und bewusst nur ein Grundstock an Entwicklerinformationen und Wegweisungen gegeben.

Die MMC ist in der Lage, eine Vielzahl an Darstellungsmöglichkeiten und Steuerelementen zu integrieren. Darüber hinaus bietet sie Schnittstellen zu neuen Verwaltungstechnologien wie beispielsweise der Windows Management Instrumentations (WMI) an.

Somit sei der Leser dieser Arbeit an dieser Stelle auf die Befehlsreferenz des Microsoft Developer Networks [MsLP04] verwiesen. Der Umfang und die Qualität der Informationen haben sich im Laufe der Erstellung dieser Diplomarbeit stark verbessert und bilden mittlerweile eine umfassende Ressource, auf die man bei der Entwicklung eines MMC Projekts nicht verzichten sollte.

5 Referenzen

5.1 Literatur

[Bus98] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: „Pattern-orientierte Softwarearchitektur"; Addison-Wesley 1998; ISBN 3-8273-1282-5

[Gri99] Grimes, R., Reilly, G., Stockton, A., Templeman, J., Watson, K.: „Beginning ATL 3 COM Programming"; Wrox Press Inc 1999, Programmer to Programmer;
ISBN 1-861001-20-7

[Han98] Hanner, K.: „Analyse und Archivierung von Benutzerberechtigungen in einem Windows NT Netzwerk"; Diplomarbeit, Johannes Kepler Universität Linz 1998

[Ise00] Iseminger, D., Microsoft Press: „Active Directory Services for Windows 2000 – Technical Reference"; Microsoft Press 2000; ISBN 0-7356-0624-2

[McCl03] McClure, St., Scambray, J., Kurtz, G.: „Das Anti-Hacker-Buch"; mitp-Verlag 2003; ISBN 3-8266-1375-9

[Mmc00] Microsoft Press: „Management Console Design and Development Kit"; Microsoft Press 2000; ISBN 0-7356-1038-X

[Sed92] Sedgewick, R.: „Algorithmen"; Addison-Wesley 1992;
ISBN 3-89319-402-9

[Tan00] Tanenbaum, A. S.: „Computernetzwerke"; Pearson Studium 2000;
ISBN 3-8273-7011-6

[W2kA00] Microsoft Press: „MCSE Training Kit - Windows 2000 Active Directory Services”; Microsoft Press 2000; ISBN 0-7356-0999-3

[W2kS00] Microsoft Press: „Windows 2000 Security”; Microsoft Press 2000; ISBN 0-7356-0858-X

5.2 Internet

[Ach02] Achleitner, M.: „Security in Windows 2000/XP für NTFS und Registry“; Vortrag zur Diplomarbeit; Johannes Kepler Universität Linz; 2002 http://www.fim.uni-linz.ac.at/sat/SAT2_NTFS_Registry.pdf

[Bol00] MSDN Magazine, Boldt, T.: „MMC: Designing TView, a System Information Viewer MMC Snap-in“; Artikel vom Dezember 2000 http://msdn.microsoft.com/msdnmag/issues/1200/tview/default.aspx

[CoPr01] The Code Project Internet Forum, Finker, L.: „Developing MMC Snap-Ins using ATL”; Forumsartikel vom Juni 2001 http://www.codeproject.com/atl/mmcsnap.asp

[Hel00] Helml, T.: „Darstellung von Rechtestrukturen in Verzeichnisdiensten am Beispiel Active Diretory“; Diplomarbeit; Johannes Kepler Universität Linz; 2000 http://www.fim.uni-linz.ac.at/diplomarbeiten/diplomarbeit_helml/ diplomarbeit_helml.pdf

[Hör99] Hörmanseder, R., Hanner, K.: „Managing Windows NT file system permissions (A security tool to master the complexity of Microsoft Windows NT file system permissions)“; Publikation im Journal of Network and Computer Applications; 1999 http://www.fim.uni-linz.ac.at/sat/sat_description.htm

[Hör00] 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

[Msdn04] Microsoft Developer Network Library; *letzter Zugriff: Oktober 2004*
http://msdn.microsoft.com/library/

[MsLN04] MSDN Library, MMC Notifications; *letzter Zugriff: Oktober 2004*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mmc/mmc/mmcn_menu_btnclick.asp

[MsLP04] MSDN Library, Platform SDK, Microsoft Management Console; *letzter Zugriff: Oktober 2004*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mmc/mmc/microsoft_management_console_start_page.asp

[MsKb04] Microsoft Knowledge Base Article - 243330; Artikel vom September 2004
http://support.microsoft.com/default.aspx?scid=kb;en-us;q243330

[Mwhp99] MSDN Library, Microsoft Management Console: Overview, White Paper; Dokument vom Oktober 1999
http://www.microsoft.com/windows2000/docs/MMCover.doc

[Psdk03] Microsoft Downloads, Microsoft Platform Software Development Kit; aktuellste Version: Februar 2003
http://www.microsoft.com/msdownload/platformsdk/sdkupdate/psdk-full.htm

[Sat04] Projekt: Security Analysis Tool (SAT), Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM), Johannes Kepler Universität Linz; *letzter Zugriff: Oktober 2004*
http://www.fim.uni-linz.ac.at/sat

[Zar02] Zarda, G.: „Visualisierung von Rechtestrukturen in Windows 2000 Netzwerken"; Diplomarbeit; Johannes Kepler Universität Linz; 2002
http://www.fim.uni-linz.ac.at/diplomarbeiten/diplomarbeit_zarda/ Visualisierung von Rechtestrukturen in Win2k Netzwerken.pdf

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.

....

Heinrich Schmitzberger

7 Lebenslauf

Persönliche Daten

Heinrich Schmitzberger
4040 - Linz, Voltastr. 5

Tel.: +43 699 11961576
E-Mail: hs@mcmedia.at

Ausbildung

1985-1989	Volksschule Frankenburg am Hausruck
1989-1993	Bundesgymnasium Ried im Innkreis
1993-1997	Bundesrealgymnasium Ried im Innkreis; Schwerpunkt Informatik Matura im Juni 1997
1997-1998	Absolvierung des Präsenzdienstes in Salzburg
seit WS 98/99	Studium der Informatik an der J.K. Universität Linz

Studienbegleitende Tätigkeiten

seit WS 99/00	Entwicklung von Webapplikationen in Kooperation mit diversen Werbeagenturen und Designfirmen
seit WS 03/04	Aufnahme der selbständigen Tätigkeit, Gewerbe: Dienstleistungen in der automatischen Datenverarbeitung und Informationstechnik

Besondere Kenntnisse

Programmiersprachen	C / C++, Java, Pascal / Modula, Basic, C#
Webprogrammierung	PHP, Java Servlets / JSP, Perl, ASP, Java Script
Datenbanken	Erfahrung in der Entwicklung und Umsetzung von Datenbank-applikationen, Plattformen: Microsoft SQL Server, MySQL, Oracle
Administration	Erfahrung in der professionellen Serveradministration von Linux Servern, Sicherheitsmanagement; Kenntnisse im Shell Scripting
Sprachkenntnisse	Englisch Französisch (Maturaniveau) Spanisch (Grundkenntnisse)