Thema :

Zentrale Benutzerverwaltung und Zugangskontrolle in einer heterogenen Systemumgebung : Analyse und Integration von AccessMaster

Diplomarbeit

vorgelegt an der Fachhochschule Köln Abteilung Gummersbach

ausgearbeitet von : Thomas Kriener

1. Prüfer : Prof. Dr. Horst Stenzel
2. Prüfer : Prof. Dr. Frank Victor

Gummersbach im Juni 1998

In dieser Diplomarbeit habe ich das Produkt AccessMaster aus dem Produktportfolio OpenMaster der Groupe Bull analysiert und die notwendigen Vorbereitungen getroffen, um es in den Betrieb des Labors für allgemeine Datenverarbeitung der Fachhochschule Köln, Abteilung Gummersbach zu integrieren. Die dafür erforderlichen Softwareentwicklungen habe ich soweit durchgeführt, daß die wesentliche Funktionalität gegeben ist.

An dieser Stelle möchte ich mich auch bei meinen Kollegen des OpenMaster Teams und den Laboringenieuren des Labors für Allgemeine Datenverarbeitung bedanken. Ganz besonderer Dank gilt Herrn Christian Gansberg und Herrn Dr. Uwe Fasting, die bei Fragen und Tests stets zur Verfügung standen.

Inhaltsverzeichnis

Vorteile einer zentralen Benutzerverwaltung und Zugangskontrolle

Eine kurze, aber präzise Zusammenfassung der Vorteile einer zentralen Benutzerverwaltung und Zugangskontrolle ist im AccessMaster White Paper (Seite 6,7) und im folgenden sinngemäß ins Deutsche übersetzt. Die Darstellungen sind auch ähnlich in den Untersuchungen der InformationWeek („Bull : The Secret Is Out“ und „InformationWeek Labs „Virtual Enterprises““) zu lesen:

Heutige IT-Strukturen gehen immer mehr von Zentralrechnern hin zu verteilten Client-Server Anwendungen. Die Informationen sind über mehrere Systeme verteilt :

- *Mainframes*
- *Abteilungsserver*
- *Netzwerk Betriebssysteme*

Zusätzlich sind die Systeme (und deren Informationen) oft über mehrere Standorte, die über Weitverkehrsnetze oder Intranets verbunden sind, verteilt. Dabei gibt es viele Möglichkeiten auf die Informationen zuzugreifen. Arbeitsplatz PCs und Laptops können auf diese Systeme

- *von einem Büro an jedem Standort aus*
- *durch einen Einwählrouter über das Telefonnetz*
- *aus dem Internet*

zugreifen.

In solch verteilten Client-Server Umgebungen ist das Sicherheitskonzept meist inkonsistent, fragmentiert und unvollständig:

- *Security-Policies, die die Zuständigkeiten und Zugriffsregeln einzelner Benutzer in Abhängigkeit ihrer Tätigkeit definieren, werden nur unvollständig durchgesetzt, wenn sie überhaupt existieren.*

- *Zugriffsregelungen werden von Mitarbeitern verwaltet, die nicht für die Sicherheit verantwortlich sind.*
- *Es dauert sehr lange bis neue Mitarbeiter zugriff auf alle Daten haben, die sie für ihre Tätigkeit benötigen.*
- *Jeder Benutzer hat eine Vielzahl von Logins und Paßwörtern um seine tägliche Arbeit zu erledigen.*
- *Helpdesks verschwenden bis zu 50% ihrer Zeit mit Paßwortproblemen.*

Diese ineffiziente Sicherheitsverwaltung, zusammen mit der Benutzung offener Netze, führt zu einer Gefährdung der zu schützenden Informationen.

- *Benutzer haben Zugriff auf Informationen, die sie nicht benötigen oder sogar auf vertrauliche Informationen, von denen sie keine Kenntnis erlangen sollen.*
- *Ehemalige Benutzer haben weiterhin unbekannte Logins auf Systemen, so daß sie auch weiterhin an sensitive Daten heran kommen können.*

Eine Zentrale Benutzerverwaltung und Zugangskontrolle soll nun genau diese Probleme lösen :

- Jeder Benutzer benötigt nur noch ein Login und ein Paßwort.
- Die Verwaltung der Logins für verschiedenen Servern und Applikationen erfolgt zentral.
- Der Systemadministrator hat nichts mehr mit den Zugangsberechtigungen zum System zu tun.

Auch für das Labor für Allgemeine Datenverarbeitung der Fachhochschule Köln bietet sich eine solche Lösung an. Es sind mehrere hundert Benutzer auf mehreren Servern von wenigen Mitarbeitern zu verwalten. Hinzu kommt, daß die Fluktuation der Mitarbeiter (in diesem Fall Studenten) hoch ist. Es ist also wichtig, daß man Zugangsberechtigungen ehemaliger Mitarbeiter und Studenten vollständig von allen Systemen entfernt.

Ein weiteres Problem, das so nicht in vielen Organisationen der Fall ist, ist die Generierung von über hundert Benutzern mit Zugang zu mehreren Servern in den ersten Tagen eines neuen Semesters. Hier kann eine zentrale Zugangskontrolle ihre Vorteile voll ausspielen, denn jeder Benutzer wird nur einmal mit allen Daten erfaßt. Danach werden nur noch die für das Zielsystem spezifischen Daten, wie z.B. Homedirectory und Shell, benötigt, wenn sie von den Default-Werten, die das System vorschlägt, abweichen sollten. Der Benutzername und die UserID werden aus den Benutzerdaten übernommen.

Bestandsaufnahme

In diesem Abschnitt werden die verschiedenen Server und Clients des ADV-Labors und angrenzender Labore aufgenommen, sowie die Kommunikationsbeziehungen analysiert.

Server

IBM RS/6000 SMP J30/6 mit AIX 4.1.5 (**ADVM2**), ca. 570 Benutzer
1024 MB Hauptspeicher, 13,50 GB Plattenspeicher
Dienste mit Logins und Paßwörtern der /etc/passwd :

- POP3
- Radius für Ascend Max 4000
- Telnet
- FTP

IBM RS/6000 C10 mit AIX 4.1.5 (**CIPS2**), ca. 510 Benutzer
256 MB Hauptspeicher, 16,30 GB Plattenspeicher
Dienste mit Logins und Paßwörtern der /etc/passwd :

- POP3
- Telnet
- FTP

Sonstige Applikationen mit Paßwörtern :

- Oracle

DEC DECsystem 5500 mit Ultrix 4.4 (**ADVS2**), ca. 390 Benutzer
96 MB Hauptspeicher, 11,37 GB Plattenspeicher
Dienste mit Logins und Paßwörtern der /etc/passwd :

- POP3
- Telnet
- FTP

IBM RS/6000 C20 mit AIX 4.1.5 (**ADVS3**), keine Systembenutzer
256 MB Hauptspeicher, 28,00 GB Plattenspeicher
Applikationen mit Paßwörtern :

- SAP/R3

PC Pentium Pro mit NT 3.51 (**CI2S1**), ca. 10 Benutzer
128 MB Hauptspeicher, 8 GB Plattenspeicher
Primary Domain-Controller und Fileserver für die NT-Workstations
In Zukunft soll jeder Student einen NT-Benutzer mit lesendem Zugriff auf die lokalen Filesysteme der Workstation und die Filesysteme des Servers haben.

Einwählrouter Ascend Max 4000 (**lan-access1**), Anzahl der Benutzer wie bei ADVM2. Die Benutzer werden bei der Einwahl über RADIUS auf der ADVM2 überprüft.

Workstations

9 SGI O2 mit Irix 6.3

3 SGI Indy R5000 mit Irix 6.2

1 SGI Indigo2 Maximum Impact mit Irix 6.2 und lokalen Benutzern

4 DEC DECsystem 5000-120 mit Ultrix 4.4

1 SUN-kompatible Sparc 10/51 mit Solaris 2.3

16 NT-Workstations mit NT 3.51

ca. 40 passive Terminals die über Terminalserver mit Telnet auf die verschiedenen Unix-Server zugreifen.

In der FH und bei Studenten zu Hause gibt es viele Workstations mit allen möglichen Betriebssystemen, die mit den Systemen des ADV-Labors kommunizieren müssen. Hier wird es nicht möglich sein alle Betriebssysteme

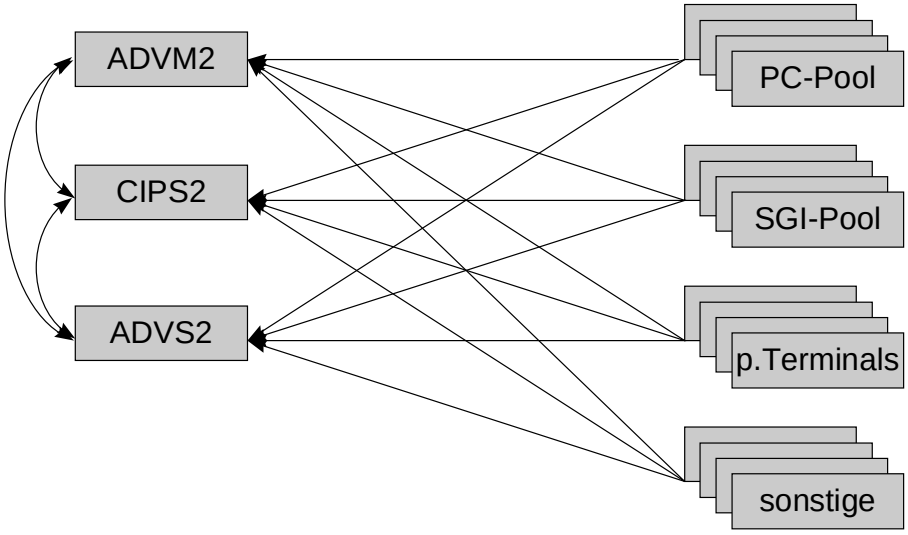
zu unterstützen, bzw. den Studenten vorzuschreiben eine bestimmte Software auf ihren Systemen zu installieren oder sogar zu kaufen.

Kommunikationsbeziehungen

Die sonstigen Systeme umfassen PCs und Workstations, die sowohl aus der Fachhochschule, dem Internet oder von zu Hause aus auf die Server zugreifen. Die 4 DECsystem 5000-120 und die SUN-kompatible Sparc werden in den folgenden Darstellungen auch zu den sonstigen Systemen gezählt.

Telnet

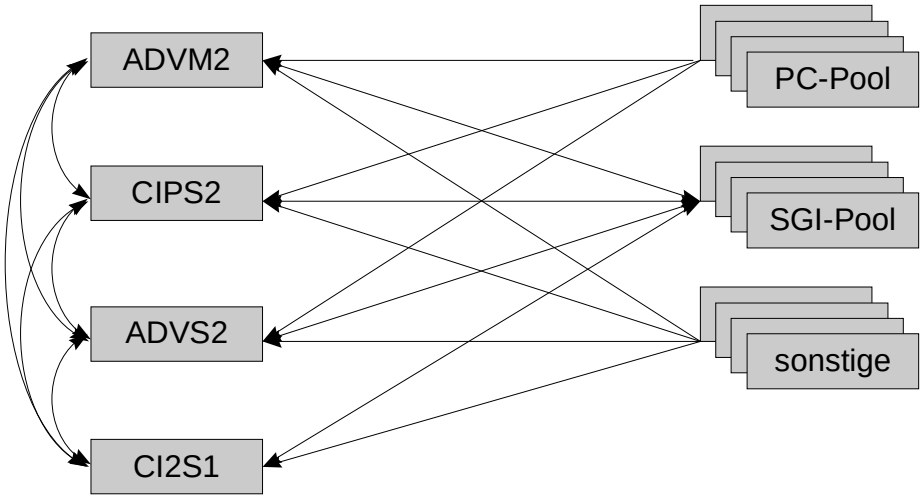
Der Telnet-Dienst wird sowohl zwischen den Unix-Systemen, als auch von allen anderen Systemen zu den Unix-Systemen genutzt.



p. Terminals = passive Terminals

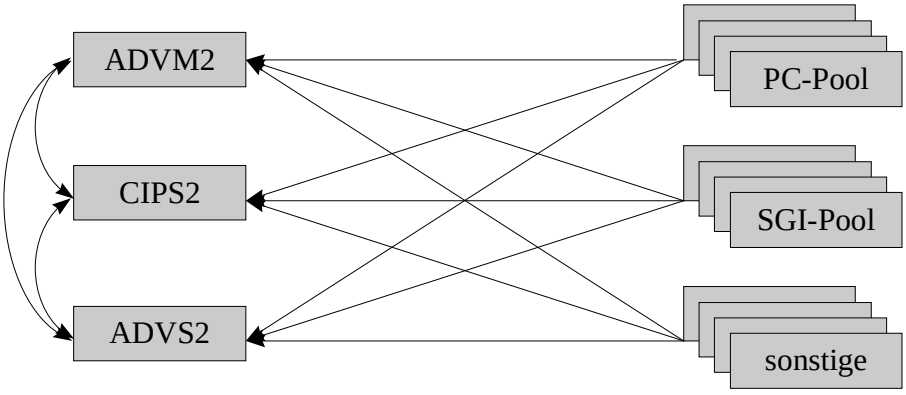
FTP

Die Kommunikationsbeziehungen des FTP-Dienstes sehen denen des Telnet-Dienstes sehr ähnlich, die passiven Terminals nutzen diesen Dienst nicht. Zusätzlich ist noch ein FTP auf den SGI-Pool und den NT-Server möglich.



POP3

Die Verwendung des POP3-Dienstes beschränkt sich auf das Abholender Mails von den Unix-Servern.



Sonstige

Die Benutzer für den Netzwerkzugang über die Ascend werden mit Radius auf der ADVM2 gegen die dortige /etc/passwd authentifiziert.

Der PC-Pool greift über das SMB-Protokoll auf den NT-Server CI2S1 zu und benutzt dortige Fileshares und Drucker.

Für die SGI-Workstations kommen die Benutzerkennungen mit NIS von der ADVM2. Das Home- und das Mail-Directory werden ebenfalls von dort aus per NFS gemountet.

Die Benutzer von Oracle und SAP/R3 werden in dieser Diplomarbeit nicht betrachtet. Für Oracle wäre eine Programmentwicklung notwendig, für die detaillierte Kenntnisse der Oracle-Benutzerverwaltung notwendig sind. Bei SAP/R3 kann man die Benutzer zur Zeit noch nicht durch externe Programme verwalten, da es bisher keine Schnittstelle gibt. Diese soll erst im Herbst 1998 von SAP bekannt gegeben werden.

Anforderungen an eine zentrale Benutzerverwaltung

Dieser Abschnitt enthält Anforderungen an eine zentrale Benutzerverwaltung für das ADV-Labor der Fachhochschule Köln.

- ***Zentrales Repository für alle Zugangsberechtigungen***

  Alle Benutzerkennungen sollen durch eine zentrale Datenbank verwaltet werden.

- ***Der Übergang muß schrittweise erfolgen***

  Es muß möglich sein, daß man die Benutzer einzeln, also Schritt für Schritt in das Benutzerverwaltungskonzept übernimmt.

- ***Workstations ohne Zusatzsoftware müssen arbeiten können***

  Alle Workstations müssen, auch ohne Zusatzsoftware, mit FTP, Telnet und POP3 arbeiten können. Diese Forderung ist auf der einen Seite durch die Anzahl der Systeme im Hause begründet, zum anderen soll es aber auch weiterhin möglich sein, von beliebigen Workstations zu Hause oder aus dem Internet heraus auf die Systeme des ADV-Labors zuzugreifen.

  Auf der anderen Seite werden dadurch die Lizenzkosten für die Zusatzsoftware erheblich gesenkt und die Einbindung neuer Betriebssysteme vereinfacht.

- ***Nutzung von Single Sign-On auf Unix-Servern***

  Benutzer sollen auf Unix-Servern in der Lage sein, einen FTP von dem System aus, auf dem sie eingeloggt sind, auf ein anderes System, das von AccessMaster verwaltet wird, ohne die Eingabe eines Paßworts durchzuführen. Das Paßwort soll automatisch durch Single Sign-On, kurz SSO, bereitgestellt werden.

Für Telnet wäre ein solche Möglichkeit ebenfalls von Vorteil, aber nicht unbedingt zwingend, da es die Möglichkeit gibt „rlogin“ und „rsh“ zu nutzen.

- ***Benutzer haben verschiedene Logins auf verschiedenen Maschinen***

  Der Benutzer mit dem primary Login A hat z.B. auf einem System das secondary Login X und auf einem anderen System das secondary Login Y. Die Benutzerverwaltung muß in der Lage sein diesen Zustand zu verwalten. Für die Zukunft ist allerdings anzustreben, daß die Benutzer auf allen Maschinen die gleichen Logins haben. Dies hat insbesondere den Vorteil, daß man später die Home-Directories automatisch per NFS von einem zentralen Fileserver mounten kann.

- ***Die Einwähl-Authentisierung soll integriert sein***

  Die Authentisierung gegenüber dem Einwähl-Router soll mit dem primary Login und Paßwort erfolgen.

- ***Benutzer sollen aus Textdateien importierbar sein***

  Da es zu Beginn des Semesters eine große Anzahl neuer Benutzer gibt, die in kurzer Zeit angelegt werden müssen, soll es die Möglichkeit geben die Benutzer aus einer Textdatei in das zentrale Repository zu importieren.

- ***Benutzer sollen auf allen Systemen gleichzeitig löschbar sein***

  Die Benutzerkennungen eines Benutzers sollen mit einem Schritt von allen Systemen gelöscht werden können. Dies vereinfacht die Administration, wenn ein Mitarbeiter oder Student die Fachhochschule verläßt.

- ***Die Administration soll verteilbar sein***

  Da die Systeme von unterschiedlichen Administratoren mit verschiedenen Aufgaben betreut werden, soll die Administration der Benutzerrechte verteilbar, jedoch zentral überprüfbar sein.

Vergleich von AccessMaster mit Alternativen

In diesem Abschnitt sind einige Alternativen zu AccessMaster zusammengetragen und kurz deren Leistung beschrieben. Auf Grund der Aufgabenstellung und Mangels an Testsystemen kann es sich hierbei nicht um eine Marktanalyse handeln. Als Grundlage dienen die allgemeinen Marketing-Unterlagen der jeweiligen Webserver der Firmen.

Möglichkeiten

In der Auswahl der Möglichkeiten ist auf die Marktführer von Integrierten Management-Plattformen beschränkt, also direkte Konkurrenten von OpenMaster insgesamt.

Eigenentwicklung

Eine Eigenentwicklung scheidet meiner Meinung nach aus, da der Aufwand für die Programmierung und die spätere Wartung in einer heterogenen Systemumgebung zu groß ist. Ganz abgesehen davon wird es schwer sein, innerhalb der Fachhochschule Fachleute für alle verwendeten Betriebssysteme zu finden, die das Know-How und die Zeit haben ein solches System konzeptionell und programmiertechnisch umzusetzen.

Es wäre notwendig zuerst ein Konzept zu entwickeln, bei dem die folgenden Fragen zu klären sind

- Was muß im zentralen Datenrepository enthalten sein?
- Welches Datenbankmodell ist angemessen?
- Welche Betriebssysteme und Applikationen sollen unterstützt werden?
- Wie werden die Daten auf die zu managenden Systeme verteilt?
- Wie können bestehende Daten importiert werden?
- Wie kann Single Sign-On verwirklicht werden?

- Wie kann das Audit erfolgen?
- Wie wird die Bedienungsoberfläche gestaltet?

Nachdem diese Fragen geklärt sind beginnt eigentlich erst die Hauptarbeit, die Implementation für die verschiedenen Plattformen.

Zu guter Letzt darf man auch den Aufwand für die Wartung des ganzen Systems nicht unterschätzen, denn ständig erscheinen neue Versionen der unterstützten Plattformen, die dann wieder neu integriert werden müssen.

HP OpenView : IT/Administration

In HP OpenView ist ein User-Management integriert, das allerdings nur Unix- und NetWare-Zielsysteme steuern kann.

Die unterstützten Zielsysteme sind :

- HP-UX
- Sun Solaris
- IBM AIX
- Novell NetWare

Für die Datenhaltung wird Oracle eingesetzt. Die Management-Plattform läuft auf HP-UX.

Es ist lediglich die Benutzerverwaltung vorgesehen, SSO ist nur außerhalb von OpenView in dem Produkt HP Praesidium Single Sign-On verfügbar. Inwieweit eine Integration von HP Praesidium und OpenView existiert ist mir nicht bekannt. Die Administration in OpenView ist delegierbar, so daß Administratoren das Recht erhalten können Teile der Organisation zu verwalten.

Für nicht unterstützte Unix-Plattformen könnte man die NIS-Unterstützung nutzen, zu der ich keine weiteren Informationen zu Art und Umfang der Unterstützung gefunden habe.

Tivoli TME 10 User Administration

Tivoli hat in seinem Framework ein Produkt namens „User Administration“. Dieses Produkt kann Benutzer auf folgenden Zielsystemen verwalten :

- Sun OS, Sun Solaris
- HP-UX
- IBM AIX
- Windows NT
- NetWare
- OS/390 (in Verbindung mit Tivoli Global Enterprise Manager)
- DEC UNIX
- NCR 3000
- Sequent
- SGI IRIX
- DG-UX
- Intel
- SCO UnixWare
- Siemens Pyramid

Als Management-Plattformen kommen folgende in Frage :

- Sun OS, Sun Solaris
- HP-UX
- IBM AIX
- Windows NT

Die Administration ist auch bei diesem Tool auf verschiedene Administratoren mit verschiedenen Aufgaben und Rechten verteilbar. Single Sign-On ist bei Tivoli nicht verfügbar. NIS wird unterstützt, indem die Datenbanken auf dem NIS Master gepflegt werden.

CA-Unicenter TNG

Im Produktportfolio von CA-Unicenter TNG gibt es zwei Applikationen, die für die Benutzerverwaltung genutzt werden können : Directory Management und SSO.

Directory Management dient der Benutzerverwaltung. Unterstützt werden folgende Umgebungen :

- Microsoft Windows NT SAM
- Microsoft Exchange Server
- Lotus Notes
- Novell NDS
- Unix NIS und NIS+

CA hat für Mainframes weitere Produkte. Da die Fachhochschule keine Mainframes betreibt, wird auf eine Betrachtung verzichtet.

Für die Datenhaltung kommen folgende Datenbanken in Frage :

- OpenIngres
- Microsoft SQL Server
- Oracle
- Sybase
- Informix

Eine Delegation der Administration ist möglich.

CA-Unicenter/SSO ist für Single Sign-On und die Zugangskontrolle zur Workstation ist zuständig.

Es werden folgende Server-Plattformen unterstützt :

- UNIX
- Windows NT

Der Client kann unter folgenden Umgebungen eingesetzt werden :

- Microsoft Windows
- Windows 95
- Windows NT

Eine Integration von Single Sign-On in eigene Applikationen kann durch die zur Verfügung gestellten APIs erfolgen.

AccessMaster

AccessMaster gehört zum Produktportfolio OpenMaster der Groupe Bull. Innerhalb von OpenMaster, einer integrierten Plattform für das Management von Informationssystemen, ist AccessMaster für Benutzerverwaltung und Security zuständig.

Die Serverplattform ist unter folgenden Betriebssystemen verfügbar :

- IBM AIX
- Sun Solaris
- Windows NT

Als Datenbank werden im wesentlichen Oracle und Ingres unterstützt. Für Windows NT ist ebenfalls Microsoft SQL Server vorgesehen.

Zu managende Systeme können sein :

- IBM AIX
- DEC OSF1
- DEC VMS
- GCOS (OS für Bull Mainframes)
- HP-UX
- IBM MVS
- LAN MAN X
- NetWare
- SGI IRIX
- Sun OS, Sun Solaris

- Windows NT

Weitere Betriebssysteme und Applikationen kann man mit dem Generic-Agent, welcher für verschiedene UNIX-Systeme zur Verfügung steht, integrieren.

Für die Single Sign-On Option werden folgende Clients unterstützt :

- AIX
- HP-UX
- SGI IRIX
- Sun OS, Sun Solaris
- Windows 3.x
- Windows 95
- Windows NT

Außerdem werden Einwählknoten mit RADIUS oder TACACS+ unterstützt. Die Integration in bestehende Umgebungen wird durch APIs unterstützt.

Vergleich

In einer ersten Betrachtung fällt auf, daß HP OpenView mit Abstand die wenigsten Betriebssysteme unterstützt und kein SSO bietet. Windows NT wird scheinbar ganz vernachlässigt.

Bei Tivoli TME 10 fehlt das Single Sign-On, obwohl Tivoli an sich sehr viele Plattformen unterstützt.

CA-Unicenter hat soweit alles zu bieten, was die Fachhochschule benötigt, es fehlt hier aber die Unterstützung für Einwählknoten.

Meiner Meinung nach ist AccessMaster momentan unter den Gegebenheiten der Fachhochschule Köln, Abt. Gummersbach die beste Wahl. Es gibt einen generischen Agenten, mit dem auch nicht unterstützte Betriebssysteme bzw. Applikationen in das User Management integriert werden können. Für die Client-Seite gibt es eine API, mit der bestehende Anwendungen wie Telnet und FTP mit Single Sign-On versehen werden können. Da die Client-API auch für

UNIX-Systeme verfügbar ist, kann sie, wie später noch in den Eigenentwicklungen gezeigt wird, für die Unterstützung von Clients ohne Zusatzsoftware genutzt werden. Die vorhandene User-Management-API ermöglicht es darüber hinaus, Teile der Administration zu automatisieren.

Eine weitergehende Unterscheidung der Produkte ist mir auf Grund der zur Verfügung stehenden Zeit und fehlender Testsysteme leider nicht möglich. So kann man z.B. die Komplexität der Architektur bei heterogenen Umgebungen leicht unterschätzen. Einen Preisvergleich kann an dieser Stelle nicht gemacht werden, da es bei diesen Produkten immer auf den Einzelfall ankommt. Integrierte Management-Plattformen sind eben keine Programme, die man „von der Stange“ kauft. Man benötigt genauso während der Einführung, wie auch später die Unterstützung der Spezialisten des Herstellers, so daß der reine Kaufpreis des Produktes zur Nebensache werden kann. Die Verbreitung der Produkte ist nicht unbedingt ein ausschlaggebender Faktor in der Bewertung von Produkten, deren Verbreitung schon durch den hohen Preis eingeschränkt ist. OpenMaster hat mit Sicherheit nicht die Verbreitung, die CA Unicenter TNG hat, doch auch Sean Gallagher (InformationWeek Labs), war vor dem Test von OpenMaster sehr skeptisch und schrieb nachher in seinem Artikel (Bull: The Secret Is Out) einfach : „... I was suitably impressed.“

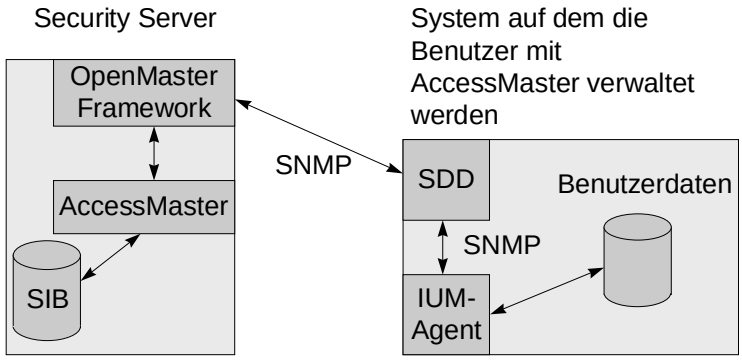
Integration von AccessMaster

Der folgende Abschnitt erläutert, wie sich AccessMaster anpassen läßt um den besonderen Anforderungen des ADV-Labors gerecht zu werden. Außerdem werden die Anforderungen an die verschiedenen Plattformen festgelegt. Im letzen Teil dieses Abschnitts werden die erforderlichen Softwareentwicklungen erläutert.

Arbeitsweise von AccessMaster

AccessMaster selbst wird auf dem sogenannten Security Server installiert. Dieser Server enthält das zentrale Repository der Zugangsberechtigungen, das bei AccessMaster als „Security Information Base“, kurz SIB, bezeichnet wird. Die Verwaltung der SIB erfolgt mit dem „I.S.M. User Management Tool“, das mit dem Befehl „ium“ gestartet wird. Dazu mehr im Abschnitt „Abbildung der Organisation“.

Die Kommunikation zwischen dem Security Server und den zu managenden Systemen erfolgt mit SNMP. Dafür wird auf dem System der „SNMP Dispatcher Daemon“ und der „Integrated User Management Agent“ benötigt. Der „SNMP Dispatcher Daemon“ bietet den Vorteil, daß er in Abhängigkeit der SNMP-OID die Requests auf mehrere Agenten verteilen kann. Der eigentliche Agent akzeptiert dann nur noch Requests von 127.0.0.1.

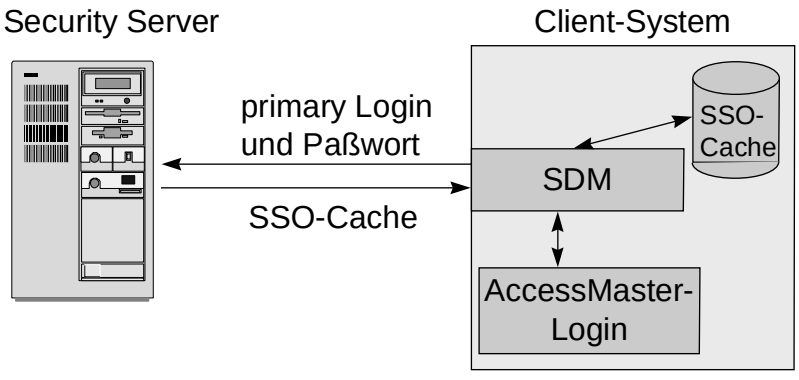


SDD : SNMP Dispatcher Daemon
IUM-Agent : Integrated User Management Agent

Da es sich bei OpenMaster um ein integriertes Managementsystem handelt, werden die Parameter für die SNMP-Kommunikation, wie Community, Portnummer und IP-Adresse, in einer Zentralen Datenbank abgelegt. Diese Datenbank befindet sich innerhalb des OpenMaster-Frameworks, auf dem AccessMaster aufsetzt.

Bei der Verwendung von Single Sign-On, kommen noch einige Komponenten hinzu. Jeder Benutzer bekommt in der SIB ein primary Login und Paßwort zugewiesen. Mit diesem Login kann sich der Benutzer gegenüber AccessMaster authentisieren. Für den Zugang zum eigentlichen System, bzw. zur Applikation, erhält der Benutzer ein secondary Login und Paßwort, das er bei der Benutzung von SSO nicht mehr kennen muß.

Der Benutzer authentisiert sich mit dem primary Login und Paßwort gegenüber AccessMaster. Danach werden die Single Sign-On Daten durch den „Security Data Manager“ vom AccessMaster in den SSO-Cache auf dem Client geladen. Damit sich der Benutzer auf Unix-Systemen mit dem primary Login anmelden kann, wird das unter Unix übliche Programm „login“ durch „loginiss“ ersetzt. Für X11 gibt es entsprechend einen AccessMaster „Secured XDM“. Unter Windows wird das Microsoft-Login durch ein AccessMaster-Login ersetzt.



Der SDM stellt eine API zur Verfügung, mit der man den SSO-Cache auslesen kann.

Für das Applikationsmanagement wird die GSS-API für DCE und Sesame unterstützt. Da die GSS-API in der FH allerdings keine Anwendung findet, wird in dieser Diplomarbeit nicht weiter darauf eingegangen.

Für das Überwachung der Benutzeraktivitäten werden von AccessMaster Audit-Daten gesammelt, die vom Auditor (eine Manager Role) ausgewertet werden können.

Abbildung der Organisation

In AccessMaster wird die Organisation unter Verwendung des X.521-Standards abgebildet. Dabei existieren zwei Sichten, eine für Personen und organisatorische Teile, und eine für Systeme und Services.

In beiden Strukturen gibt es folgende Knoten :

- Land
- Ort
- Organisation
- Organisationseinheit

In der Personensicht gibt es außerdem folgende Knoten :

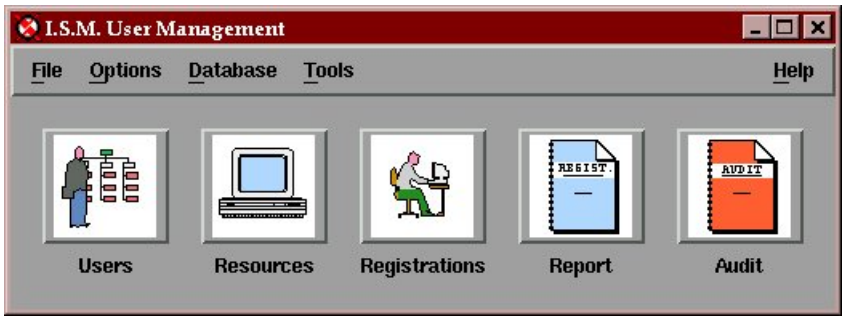
- Person
- Rolle
- Personengruppe
- Passwort-Policy
- Timetable
- Arbeitsgruppe
- Privileg
- Privilegklasse

In der Systemsicht gibt es außer den gemeinsamen Knoten :

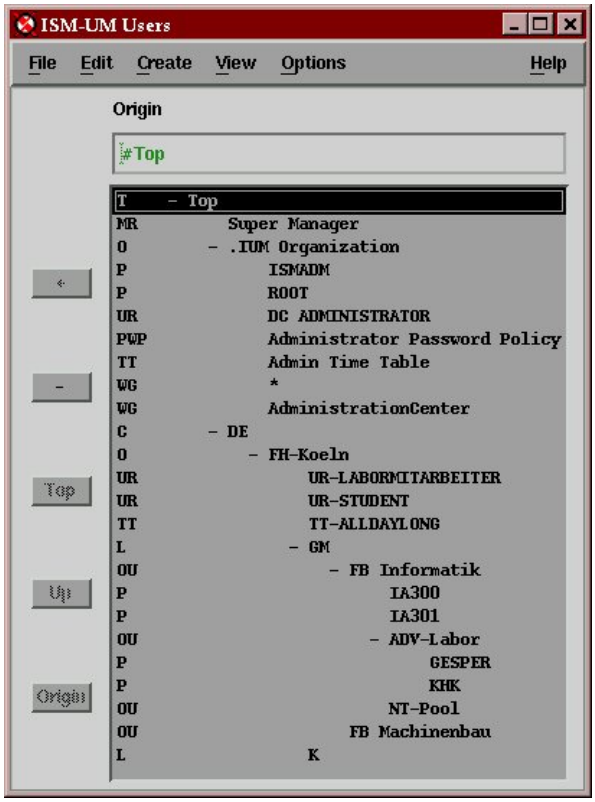
- System
- Service

- Gruppen von Services
- Applikation

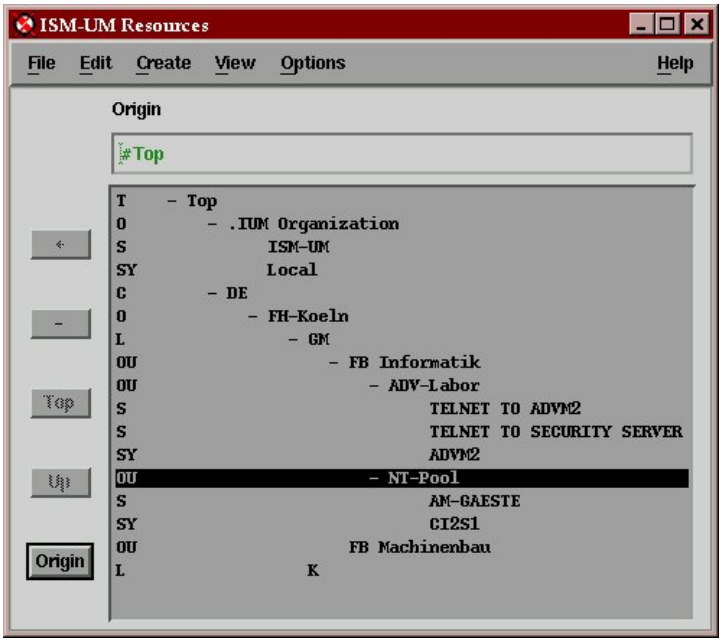
Diese Strukturen werden alle über das I.S.M. User Management Tool gepflegt und in der SIB abgespeichert.



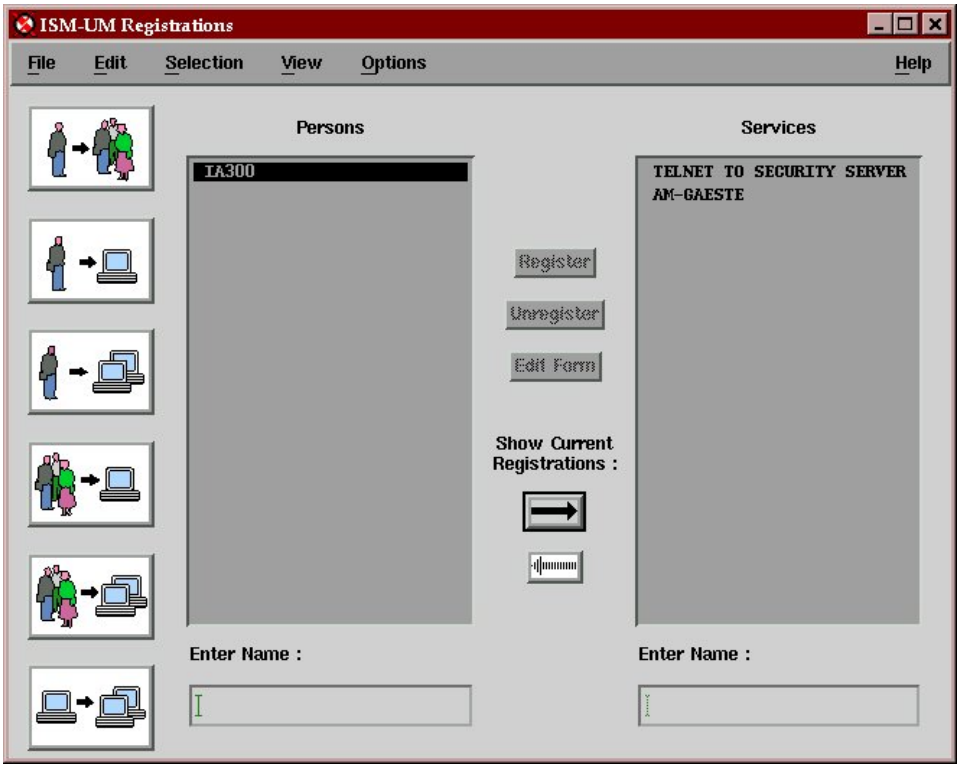
Alle gemeinsamen Elemente, sowie die Elemente der Personensicht werden mit dem Dialog hinter dem Users-Button gepflegt.



Die Systeme, Services und Applikationen werden mit dem Dialog hinter dem Resources-Button verwaltet.

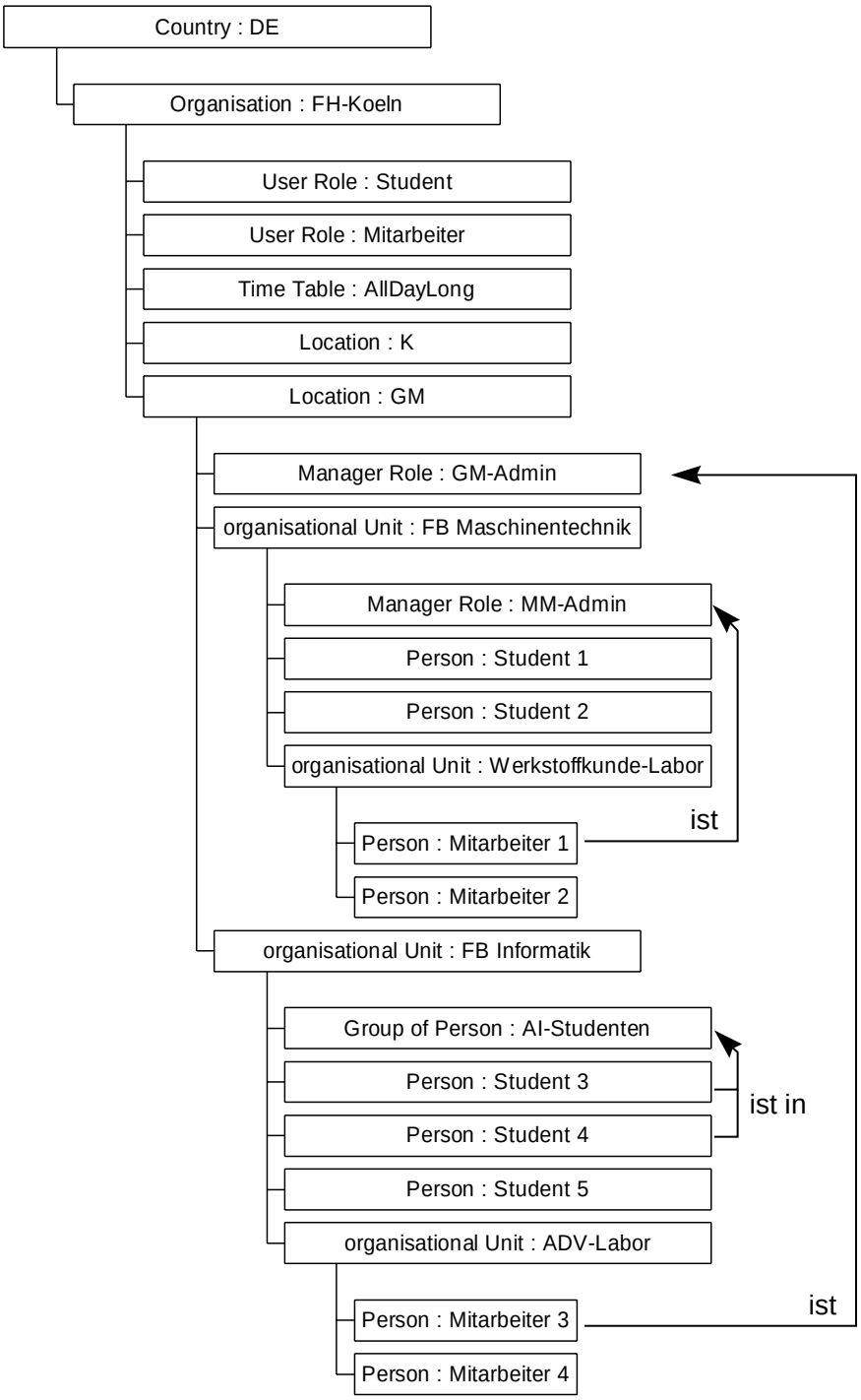


Personen werden an Services oder Gruppen über den Registrations-Button registriert.



Personen

Für die FH-Köln, Abt. Gummersbach sieht die Personensicht wie folgt aus :



Die Baumstruktur zeigt die Zugehörigkeit der Personen zu einzelnen Organisationen und Organisationsteilen. Für die Administration hat dies den Vorteil, daß man einzelnen Administratoren Rechte für Teilbäume vergeben kann, in dem man ihnen eine „Manager Role“ zuweist (GM-Admin und MM-Admin). So ist es im obigen Beispiel möglich, das der „Mitarbeiter 1“ des Werkstoffkunde-Labors (Fachbereich Maschinentechnik) Studenten und Mitarbeiter seines Fachbereichs erfaßt, ohne Zugriff auf Benutzer des anderen Fachbereichs zu haben. Dahingegen kann der „Mitarbeiter 3“ des ADV-Labors alle Objekte unterhalb von GM bearbeiten. Bei den Manager Roles gibt es verschiedene Rollen, je nachdem ob ein Administrator Benutzer erfassen bzw. registrieren oder Systeme erfassen soll.

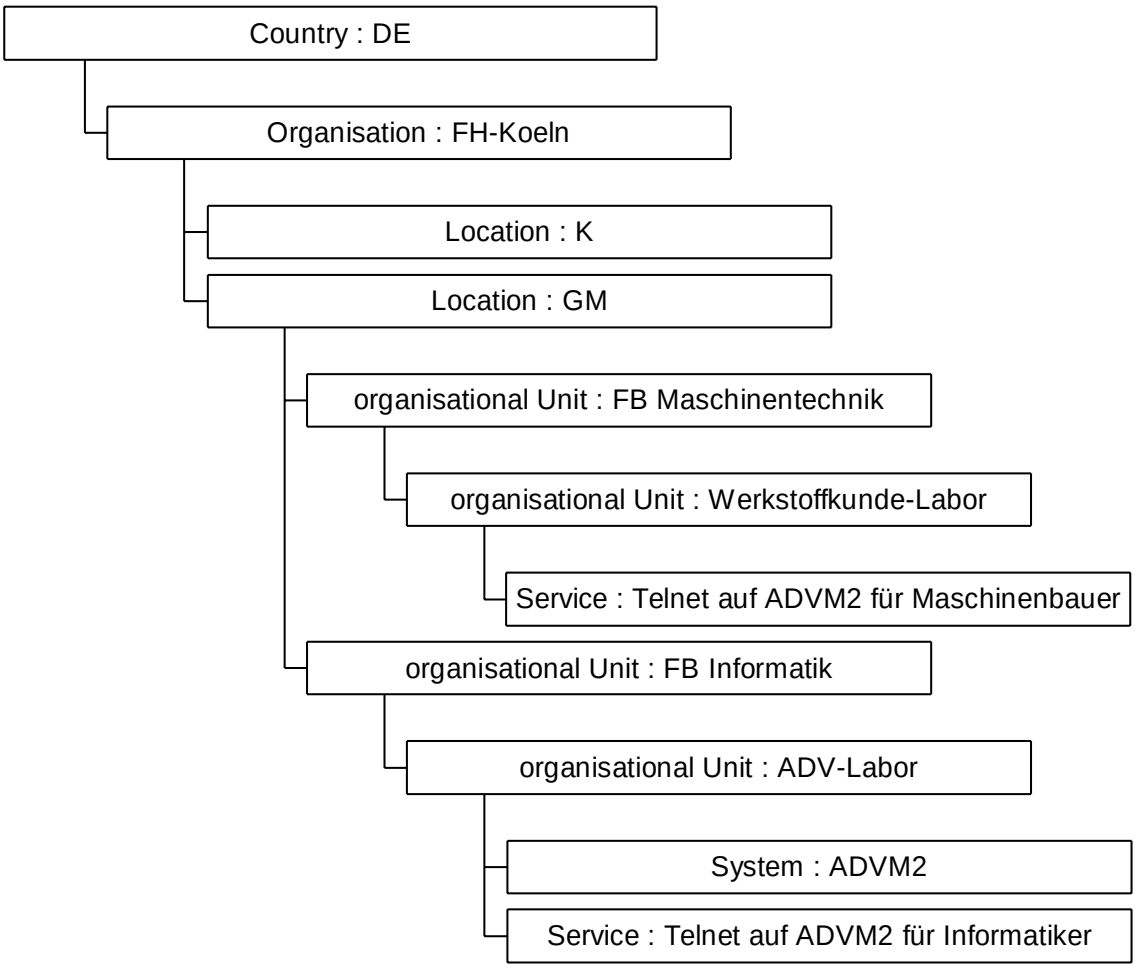
Das obige Beispiel zeigt ebenfalls, das „Student 3“ und „Student 4“ in der Gruppe „AI-Studenten“ sind. Gruppen bieten den Vorteil das man Benutzerprofile für Personengruppen nur einmal festlegen muß. Anschließend werden neue Benutzer nur noch an dieser Gruppe registriert und erhalten damit Zugriff zu allen notwendigen Systemen.

Das „Time Table“-Objekt legt fest, wann sich Benutzer gegenüber AccessMaster authentisieren können.

Jeder Benutzer hat eine „User Role“. Diese Objekte werden in der aktuellen Version für die Zuweisung von Windows-Desktops, sowie Sesame- und DCE-Parametern genutzt.

Systeme, Services und Applikationen

In dieser Ansicht werden die Ressourcen einzelnen Organisationen, bzw. Organisationsteilen zugeordnet. Durch die Verteilung der Managerrollen wird damit auch festgelegt, wer den Services neue Benutzer zuordnen kann, bzw. wer neue Systeme und Services anlegen kann.



Im obigen Beispiel kann man sehen, daß es ein System ADVM2 gibt. Auf diesem System gibt es zwei Services (bei Unix-Systemen entspricht dies den Gruppen), einen für Maschinenbauer und einen für Informatiker. Da diese Services jeweils einem anderen Labor zugeordnet sind, kann „Mitarbeiter 1“ des Werkstoffkunde-Labors Maschinenbauer für diesen Service berechtigen und „Mitarbeiter 3“ des ADV-Lavors alle Gummersbacher Studenten für alle Gummersbacher Services.

Hier sei darauf hingewiesen, daß in der Systemmaske das Feld „SSO Identifier 1“ mit dem Systemnamen zu belegen ist. Dieser Name wird später von AccessMaster-Komponenten wie „loginiss“ und dem „AccessMaster Secured XDM“ benötigt um das System zu identifizieren.

Es gibt auch bei den Services die Möglichkeit Gruppen zu bilden. Diese Gruppen haben dann den Objekt-Typ „Group of Service“. Benutzer oder Benutzergruppen können dann gleichzeitig an mehreren Services registriert werden bzw. es kann die Registrierung zu dieser Gruppe mit einem Mal aufgehoben werden.

Systemvorraussetzungen

AIX

Für die AIX-Produkte ist AIX Version größer gleich 4.1.4 oder 4.2 erforderlich.

Security Server

Der Security Server kann zum Flaschenhals bei der Authentisierung der Benutzer werden. Das OpenMaster-Support Team der Produktlinie in Frankreich hat für die Auslegung des Security Servers im „Marketing Configurator“ eine Tabelle erarbeitet.

Die Tabelle gilt unter folgenden Annahmen :

- 40% der Benutzer authentisieren sich in der Stoßzeit, 10% vorher.
- Während der Stoßzeiten findet keine massive Administration statt, wie z.B. Gruppen-Registrierungen.
- Die Hälfte der Benutzer nutzen DCE oder Sesame während der Stoßzeit.

	ISM-355	ISM-450	ISM-480	ISM-580	ISM-680
Anzahl der Workstations	bis 1000	bis 1000	bis 1000	bis 2000	bis 4000
Anzahl simultane Administratoren	4	4	4	8	15

	Prozessor	Hauptspeicher	Plattenkapazität
ISM-355	Estrella PPC 604/166 MHz	128 MB	3 GB
ISM-450	Estrella 2x PPC 604/166 MHz	128 MB	4 GB
ISM-480	ESCALA M104	128 MB	4.2 GB
ISM-580	ESCALA M204	256 MB	4.2 GB
ISM-680	ESCALA D204	512 MB	4.2 GB

Da Stoßzeiten nur zum Praktikumsbeginn zu erwarten sind, beschränkt sich die Anzahl der Benutzer maximal auf die zur Verfügung stehenden Workstations, das sind insgesamt 12 SGIs + 16 NT-Workstations + 40 passive Terminals, so daß es sich um max. 68 Benutzer handelt die sich gleichzeitig anmelden. Der kleinste Server sollte also ausreichen.

AccessMaster und das OpenMaster Framework benötigen 650 MB Plattenplatz, Oracle mindestens 500 MB.

Da die Administration über X11-Anwendungen erfolgt, sind mindestens die X11- und Motif-Bibliotheken von AIX zu installieren, auch wenn das System selbst keinen graphischen Adapter besitzt. Für Oracle sind außerdem die Mathematik-Bibliothek und das Kommando „make“ erforderlich.

AccessMaster benutzt als Datenbank Oracle, hier wird die Version 7.3.2 oder 7.3.3 benötigt.

Von OpenMaster ist das Framework Version 4.54 und AccessMaster 3.1 zu installieren. Damit AccessMaster als Radius-Server arbeiten kann, ist die NAS-Unterstützung zu installieren. Da die Benutzer auf dem Security Server idealer weise ebenfalls mit AccessMaster verwaltet werden muß auch der „SNMP Dispatcher Daemon“ und der „Integrated User Management Agent“ auf dem Security Server installiert werden.

Workgroup Server

Für Workgroup Server unter AIX gibt es keine besonderen Anforderungen an die Systemumgebung.

Von AccessMaster ist für die Benutzerverwaltung der „SNMP Dispatcher Daemon“ und der „Integrated User Management Agent“ zu installieren.

Um sich an dem System mit dem primary Login und Paßwort anmelden zu können sind der „Workgroup File Server“, der „File Transfer Server“, der „Security Data Manager“ sowie der „Secured XDM End login“ zu installieren.

Der Platzbedarf beträgt ca. 10 MB. Falls die Audit-Daten auf Grund des Netzwerk-Traffic lokal gesammelt werden, so wird dafür ebenfalls Plattenplatz benötigt, der von der Verweildauer der Daten und dem Datenaufkommen abhängt.

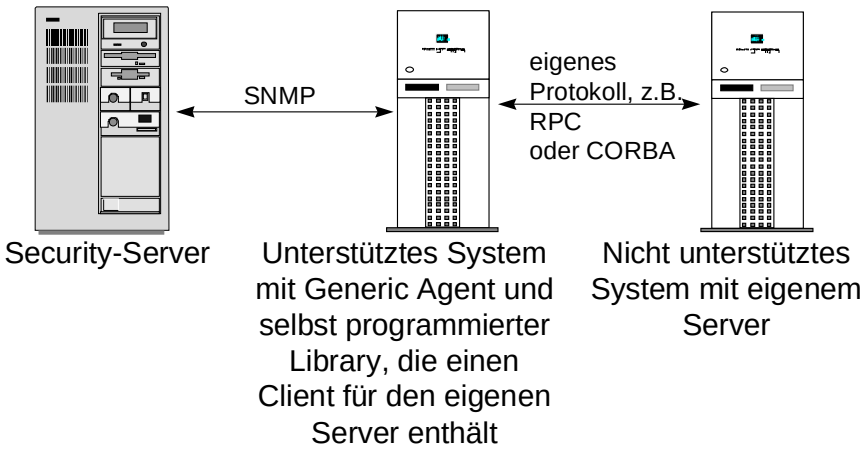
Workstation

Eine AIX Workstation, also ein AIX-System, auf dem keine Benutzer verwaltet werden, bzw. dessen Benutzer über einen NIS-Master verwaltet werden, benötigt nur den „Security Data Manager“ und den „Secured XDM End Login“.

Ultrix

Um die Benutzer auf der ADVS2 zu verwalten, müßte man mit dem Generic Agent die Benutzerverwaltung selbst programmieren, da ein Integrated User Management Agent für Ultrix 4.4 nicht zur Verfügung steht.

Der Generic Agent dient dabei als Proxy. Er wird auf einem unterstützten System, z.B. AIX, installiert und leitet die SNMP-Requests an eine zu erstellende Library weiter. Diese Library führt dann meistens RPC-Funktionsaufrufe auf das Zielsystem durch.



Der Aufwand für die Implementierung der Library und des eventuell notwendigen Servers hängt im wesentlichen von den verfügbaren Hilfsmitteln auf dem Zielsystem ab.

Der Generic Agent selbst erwartet folgende 18 Funktionen in der Library

Funktionsname	Aufgabe
Ium_GA_Init	Initialisierung der Library, wird aufgerufen wenn die Library geladen wird.
Ium_GA_GetSystem	Liefert Attribute des Systems, wird insbesondere genutzt um die Kommunikation zu testen.
Ium_GA_SetSystem	Setzt Attribute die das System beschreiben.

Ium_GA_CreateService	Erzeugt neuen Service, unter Unix entspricht dies der Erzeugung einer Gruppe
Ium_GA_DeleteService	Löscht einen Service.
Ium_GA_GetService	Liefert Attribute eines Service.
Ium_GA_SetService	Setzt Attribute eines Service.
Ium_GA_ListOfServices	Liefert eine Liste der verfügbaren Services.
Ium_GA_CreateAccount	Diese Funktion legt einen Benutzer an.
Ium_GA_DeleteAccount	Löscht einen Benutzer.
Ium_GA_GetAccount	Liefert Attribute eines Benutzers.
Ium_GA_SetAccount	Setzt Attribute eines Benutzers.
Ium_GA_ListOfAccounts	Liefert eine Liste der Benutzer.
Ium_GA_Register	Diese Funktion registriert einen Benutzer mit einem Service. Unter Unix entspricht dieser Vorgang dem zuordnen eines Benutzers zu einer Gruppe.
Ium_GA_Unregister	Entfernt die Registrierung eines Benutzers von einem Service. Unter Unix würde dies der Entfernung einer Gruppe des Benutzers entsprechen.
Ium_GA_ListOfRegistrations	Diese Funktion liefert alle Registrierungen in Paaren aus Benutzern und Services.
Ium_GA_ListOfServicesForAccount	Diese Funktion liefert die Registrierungen an Services eines Benutzers.
Ium_GA_ListOfAccountsForServices	Diese Funktion liefert die Benutzer, die an einen Service registriert sind.

Wie man am Funktionsumfang erkennen kann, lassen sich mit dieser Methode nicht nur Benutzer von Systemen verwalten sondern auch Benutzer von Applikationen, bei denen eine Authentisierung erforderlich ist, wie z.B. Datenbanken oder Warenwirtschaftssyteme.

SGI/Irix

Das Betriebssystem IRIX wird bisher in der Version 5.3 unterstützt. Da die AccessMaster-Komponenten mit System V Release 4 Sockets arbeiten ist das Paket „eoe.sw.svr4net“ nachträglich zu installieren.

In der Datei „/etc/services“ müssen die Services issft und issft-data von Hand nachgetragen werden.

```
issft-data  3782/tcp    # ISS
issft       3783/tcp    # ISS
```

Als Workstation benötigt ein SGI/IRIX-System nur den „Security Data Manager“ und den „Secured XDM End Login“.

Die SGIs im ADV-Labor der Fachhochschule Köln, Abt. Gummersbach laufen zwar unter IRIX 6.2 und 6.3, in den folgenden Betrachtungen wird aber davon ausgegangen, daß bis zur Einführung von Single Sign-On eine entsprechende Unterstützung vorhanden ist.

Windows NT

Workgroup Server

Für den Workgroup Server wird Windows NT Server Version 3.51 Servicepack 4 oder 4.0 Servicepack 3 unterstützt.

Für das User Management ist der „Windows NT Workgroup Server“ und der „ISM-UM Windows NT Agent“ erforderlich. Der „SNMP Dispatcher Daemon“ ist im „ISM-UM Windows NT Agent“-Paket enthalten.

Damit ein Benutzer an diesem System mit dem primary Login arbeiten kann, wird außerdem der „Basic Security Service for Windows NT“ benötigt.

Workstation

Die AccessMaster-Komponenten für Windows NT Workstation setzten Version 3.51 Servicepack 4 oder 4.0 Servicepack 3 vorraus.

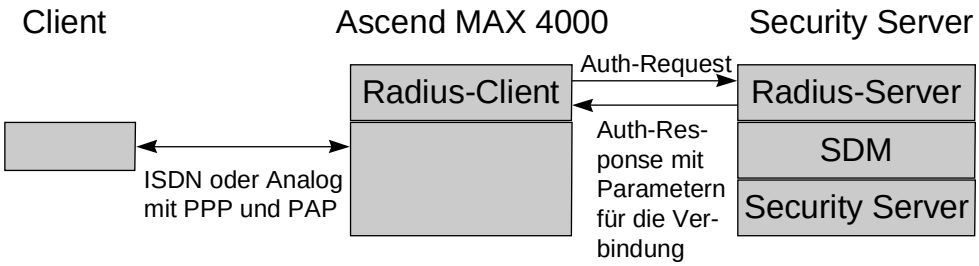
Für den primary Login wird der „Basic Security Service for Windows NT“ benötigt.

Ascend Max 4000

Für den Support von RADIUS oder TACACS+ benötigt AccessMaster die Network Access Server Unterstützung auf dem Security Server.

Für RADIUS wird außerdem das Dictionary von Ascend für die Ascend MAX 4000 benötigt, welches die herstellerabhängigen RADIUS-Felder beschreibt. In einer zusätzlichen Datei werden dann diese Parameter gegen SSO-Felder der SIB gemappt.

Die Architektur sieht dann wie folgt aus :



Softwareentwicklungen

Die folgenden Softwareentwicklungen schließen die Lücken zwischen AccessMaster und der spezifischen Umgebung des Labors für Allgemeine Datenverarbeitung an der Fachhochschule Köln, Abteilung Gummersbach. Die

Erläuterungen sind in drei Abschnitte aufgeteilt : Funktion, Konfiguration und Quelltext.

POP3-Daemon mit AccessMaster-Authentisierung

Funktion

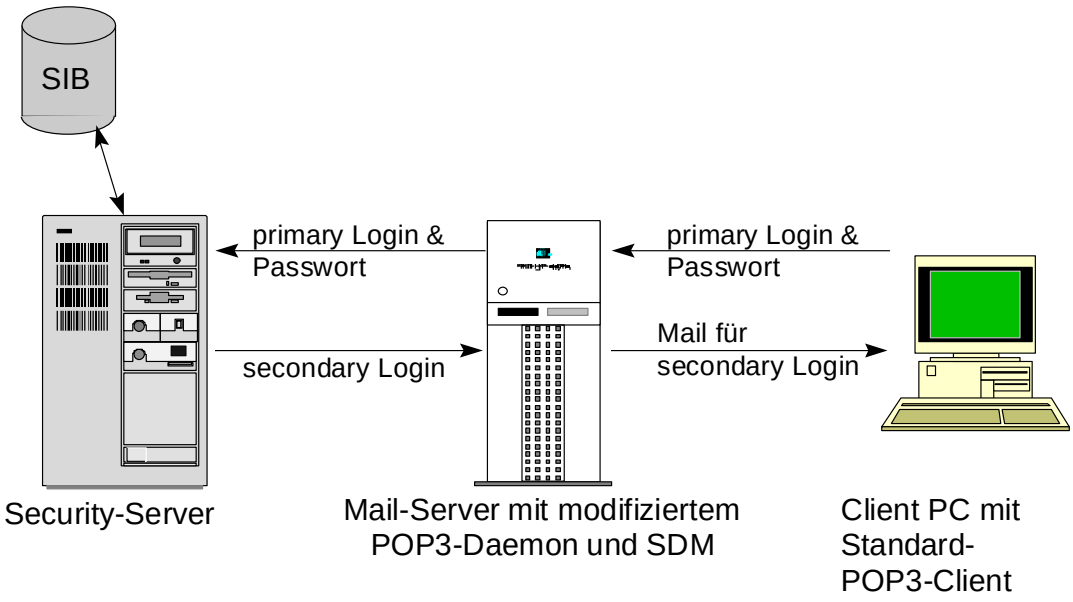
Für die Nutzung von POP3 ohne SSO-API auf dem Client mußte der POP3-Daemon geändert werden.

Der POP3-Daemon ist so geändert, daß er sich nach der Authentisierung mit dem primary Login und Paßwort das secondary Login aus den SSO-Informationen heraussucht. Um das secondary Login für das System herraus zu suchen, auf dem der POP3-Daemon installiert ist, wird der AccessMaster-Name des lokalen Systems in der inetd.conf als Parameter für den pop3d4AM angegeben.

Hier z.B. für den Security Server selbst :

```
# AccessMaster pop3-Daemon
pop3 stream tcp nowait root /ism454/new/bin/pop3d4AM pop3d4AM Local
```

Der Daemon kann gleichzeitig mit primary und secondary Logins arbeiten. Dabei ist allerdings zu beachten, das die Mengen der Login-Namen disjunkt sind. Da der Daemon zuerst versucht den Benutzer gegen AccessMaster zu authentisieren, würde sonst bei Eingabe des secondary Paßworts, für einen Login, der auch als primary Login existiert, das primary Login mit einem falschen Loginversuch belegt. Dies führt bei mehrfacher Ausführung dann evtl. zur Sperrung des primary Logins.



Quelltext

Den Quelltext des POP3-Daemons ist von BSD-Unix 4.3 übernommen und entsprechend angepaßt worden. Deshalb sind hier auch nur die Ergänzungen dokumentiert.

Ein Umstand der mir sehr gelegen kam, war der, daß es eine extra Funktion zur Überprüfung des Login und Paßworts mit Namen „verify_user“ im POP3-Daemon gibt. Dieser Funktion wird der Name und das Paßwort als Pointer übergeben, so daß das primary Login bequem durch das secondary Login auf dieser Maschine ersetzen werden kann.

Die Änderungen in der Funktion „verify_user“ sind hier fett hervorgehoben.

```
/* Verify a usercode/password */
int
verify_user(user,pass)
char *user;
char *pass;
{
      struct passwd *pwd;
#ifdef SHADOWPWD
      struct spwd *spwd;
#endif
      char *cp;
      int am_retval;
```

```
      /* First try to authenticate using AccessMaster */
      am_retval=my_login(user,pass,"");
      if(am_retval==0)
      {
            am_retval==get_systemuser(am_system,user);
            if(am_retval!=0)
                  return -1;
            my_logout();

      }
      else
      {
            /* May be it is a normal User and no AccessMaster User */
            pwd = getpwnam(user);
            if (pwd == NULL) return -1;
#ifdef SHADOWPWD
            if (!(spwd = getspnam(user)))
                  return -1;
            else
                  pwd->pw_passwd = spwd->sp_pwdp;
            if (pwd->pw_name && pwd->pw_passwd[0] == '@') {
                  if (pw_auth(pwd->pw_passwd+1, user, PW_LOGIN))
                        return -1;
                  return(setuid(pwd->pw_uid));
            }
            if (!valid(pass, pwd))
                  return -1;
            return(setuid(pwd->pw_uid));
#else
            cp = crypt(pass,pwd->pw_passwd);
            if (strcmp(cp,pwd->pw_passwd) == 0) {
                  return(setuid(pwd->pw_uid));
            } else {
                  return(-1);
            }
#endif
            return -1;
      }
}
```

Die Änderung besteht im wesentlichen darin, daß der Daemon zuerst versucht den Benutzer gegen AccessMaster zu authentisieren und falls das nicht gelingt, noch prüft, ob es sich vielleicht um einen User handelt, der nur lokal existiert.

Die Funktionen my_login und get_systemuser sind neu und in der Datei am_util.c enthalten. Die Variable „am_system“, die der Funktion „get_systemuser“ übergeben wird, ist der Parameter, der dem Daemon in der inetd.conf übergeben wird.

In der Datei main.c ist dafür diese Variable deklariert worden

```
char am_system[1024]; /* AccessMaster Name for this system */
```

und in der Datei util.c ist sie wie folgt deklariert.

```
extern char am_system[1024];
```

Die Variable wird direkt zu Beginn des Programms in der Funktion „main“ nach der Initialisierung gesetzt.

```
void main(int argc, char* argv[])
{
      int svr_state = SVR_LISTEN_STATE; /*State of POP3 server*/
      char cli_buf[CLI_BUFSIZ];          /*Buffer for client cmds*/

      initialize();

      /* Get Systemname to look for in AccessMaster */
      if(argc<2)
            fail(FAIL_AMSYSTEM_EXPECTED);
      else
            strcpy(am_system,argv[1]);

      fprintf(stdout,"+OK %s POP3 Server for AM-System %s (Version %s)
ready.\r\n",
            svr_hostname,am_system,VERSION);
      fflush(stdout);
      svr_state = SVR_AUTH_STATE;
      for ( ; ; ) {
            /* Wait for command from client */
            alarm(SVR_TIMEOUT_CLI);
            if (fgetl(cli_buf,CLI_BUFSIZ,stdin) == NULL)
                  break;
            alarm(0);

            /* Take action on client command */
            cmd_prepare(cli_buf);
#ifdef DEBUG
            if ((cli_buf[0] == 'p')||(cli_buf[0] == 'P'))
                  fprintf(logfp,"CLI: PASS\n",cli_buf);
            else
                  fprintf(logfp,"CLI: %s\n",cli_buf);
#endif
            switch(svr_state) {
            case SVR_AUTH_STATE:    /* Expecting USER command */
                  svr_state = svr_auth(svr_state,cli_buf);
                  break;
            case SVR_PASS_STATE:    /* Expecting PASS command */
                  svr_state = svr_pass(svr_state,cli_buf);
                  break;
            case SVR_TRANS_STATE:   /* Expecting mailbox command */
                  svr_state = svr_trans(svr_state,cli_buf);
                  break;
            case SVR_FOLD_STATE:    /* Need to open another mailbox */
                  svr_state = svr_fold(svr_state,cli_buf);
                  break;
            default:
```

```
                  fail(FAIL_CONFUSION);         /* Wont return */
                  break;
            }
#ifdef DEBUG
            fprintf(logfp,"SVR: %s",svr_buf);
#endif

            /* Send out response to client */
            alarm(SVR_TIMEOUT_SEND);
            fputs(svr_buf,stdout);
            fflush(stdout);
            alarm(0);
            if (ferror(stdout))
                  break;

            /* Exit when server has sent goodbye */
            if (svr_state == SVR_DONE_STATE)
                  break;
      }
fld_release();    /* [1.003] Make sure folder is released */
      exit(0);
}
```

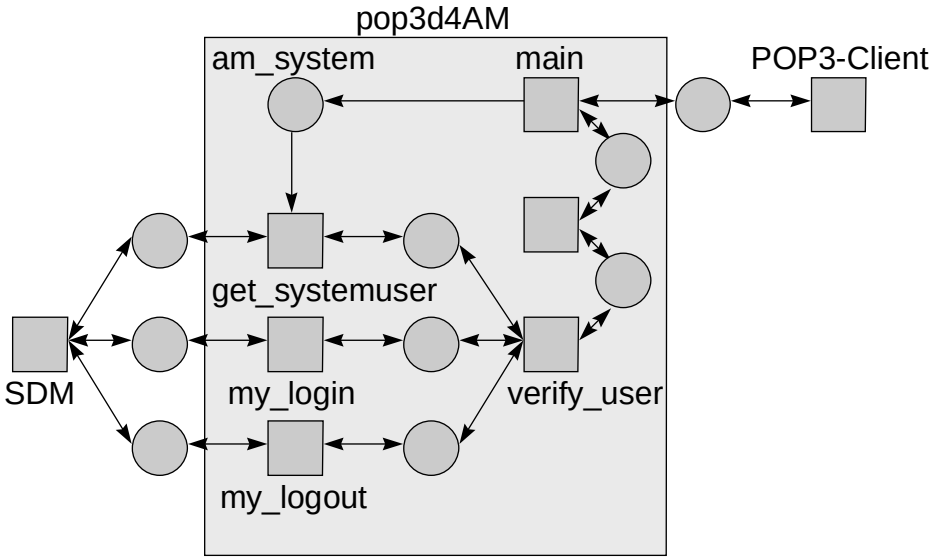
Falls der Parameter in der inetd.conf nicht gesetzt wurde, erzeugt der Aufruf „fail(FAIL_AMSYSTEM_EXPECTED);**“** die Fehlermeldung „System to look for in AccessMaster not found“.

Die Funktion „int my_login(char *user_name, char *password, char *role_name)“ nimmt den Benutzernamen, das Paßwort und Rolle des Benutzers und authentisiert den Benutzer am Security Server. Bei Erfolg liefert die Funktion „0“, ansonsten den Fehlerwert der API zurück. Wenn bei der Authentisierung kein Fehler auftritt, wird außerdem das Credential-Handle gefüllt.

Die Funktion „int my_logout (void)“ gibt das Handle wieder frei und löscht den Shared-Memory-Bereich. Damit ist der Benutzer abgemeldet.

Die Funktion „get_systemuser“ erhält als Parameter das System, auf dem für den angemeldeten Benutzer ein secondary Login existieren soll. Der Username wird in dem ebenfalls übergebenen Buffer abgelegt. Der Returnwert ist „0“ falls die Suche erfolgreich war oder den Fehlerwert der API bzw. „1“ falls kein Benutzer gefunden wurde. Da es in der SSO-API leider keine Funktion gibt mit der man nach einem System suchen kann, durchsucht diese Funktion alle Services des Benutzers bis ein Service das gesuchte System enthält.

Die im folgenden abgedruckte Datenfluß zeigt nur die ergänzten bzw. neuen Funktionen.



Programm zur Generierung von „~/.netrc“

Funktion

Für die Nutzung der SSO-Funktion von einem Unix-System aus gibt es bereits eine Datei Namens „~/.netrc“. Diese Datei wird vom FTP-Client und den r-Kommandos (rexec, rlogin und rcmd) benutzt um den Benutzer an einem entfernten System automatisch zu authentisieren.

Das Programm create_netrc ließt den AccessMaster Cache aus und fügt der Datei„~/.netrc“ Zeilen in dem Format

```
machine <Systemname> login <Benutzerkennung> password <Paßwort>
```

hinzu. Dabei wird für jede Maschine nur ein Eintrag erzeugt, so daß eventuell vorhandene Einträge ersetzt werden.

Zu Beachten ist noch, daß diese Datei im Zugriffsmodus 600 (-rw----) sein muß, damit sie von den Programmen akzeptiert wird. Existiert die Datei bisher noch nicht, so wird sie mit den korrekten Rechten angelegt.

Account- und Makro-Einträge werden ignoriert und fehlen nach der Anwendung von create_netrc.

Mir fehlt bei den Account- und Macro-Einträgen leider jede Erfahrung. Speziell bei den Accounts kenne ich auch keinen FTP-Server der diese Information anfordert bzw. verarbeitet.

Konfiguration

Eine Konfiguration ist nicht erforderlich. Ich empfehle, dieses Programm in das jeweilige Login-Skript (.profile, .cshrc oder ...) des Benutzers zu übernehmen, um die Informationen bei jedem Login zu aktualisieren.

Quelltext

Der Quelltext ist in zwei Module geteilt und in C++ geschrieben. Das Modul „create_netrc.cxx“ enthält das Hauptprogramm und die Funktion „int create_netrc(NetRC& netrc)“. Das Hauptprogramm erzeugt nur eine Instanz der Klasse „NetRC“ und übergibt sie der Funktion „int create_netrc(NetRC& netrc)“. Zum Schluß ruft es noch die Memberfunktion „void write()“ der Klasse „NetRC“ auf um die Einträge in die Datei „~/.netrc“ zu schreiben.

Die Funktion „int create_netrc(NetRC& netrc)“ liest den Cache aus und ruft mit dem secondary Login, dem secondary Paßwort und dem Maschinennamen (SSO-Identifier 1) die Funktion „void addEntry(const MachineEntry& machine)“ der Klasse „NetRC“ auf um den Eintrag hinzu zu fügen.

Das Modul „netrc.cxx“ enthält drei Klassen : MachineEntry, Token und NetRC.

Die Klasse „MachineEntry“ enthält die Daten für jeweils einen Eintrag. Sie hat nur die Funktion „void clean()“, um alle Membervariablen mit dem Leerstring zu belegen.

Die Klasse „Token“ enthält jeweils ein Token aus der Datei „~/.netrc“. Außer dem Konstruktor, der die Membervariablen mit den übergebenen Werten belegt, hat die Klasse keine Funktionen.

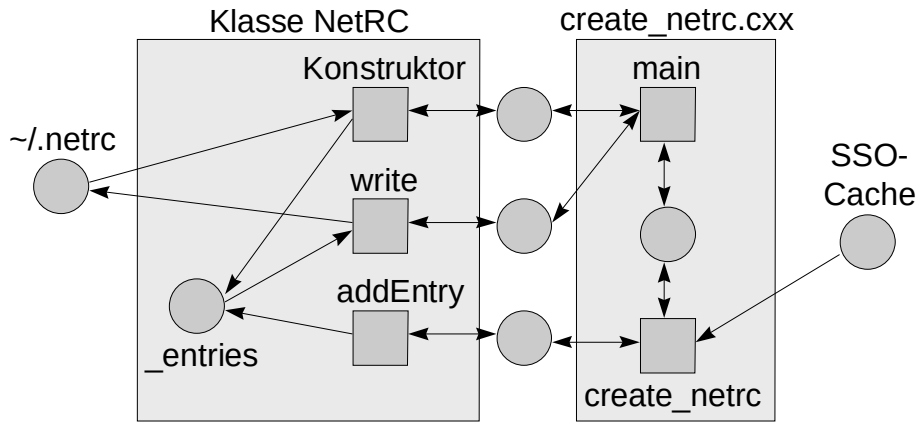
Die Klasse „NetRC“ hat die Aufgabe die Datei „~/.netrc“ auszulesen, neue Einträge entgegen zu nehmen und zum Schluß wieder wegzuschreiben. Sie enthält 6 Funktionen : den Default-Konstruktor, „void addEntry(const MachineEntry& machine)“, „int getNumEntries()“, „MachineEntry getEntry(int i)“, „void write()“, „Token getToken(ifstream& fileStream)“.

Der Default-Konstruktor liest die Datei „~/.netrc“ aus. Dazu benutzt er die Funktion „Token getToken(ifstream& fileStream)“ um die Datei in Tokens zu zerlegen. Der Code der Funktion „Token getToken(ifstream& fileStream)“ ist im wesentlichen dem in C geschriebenen FTP-Client von BSD-Unix entnommen und an C++ angepaßt worden.

Die Funktion „void addEntry(const MachineEntry& machine)“ fügt dem Vektor „_entries“ einen Eintrag hinzu, falls für diese Maschine noch keiner existiert. Der Vektor ist nicht sortiert, da es sich hier um eine sehr kleine Liste handelt.

Die Funktion „void write()“ schreibt die Einträge in die Datei „~/.netrc“.

Die Kommunikationsbeziehungen des Programms sehen wie folgt aus :



Telnet-Proxy für Kommandozeile

Funktion

Für Telnet gibt es leider kein Standardverfahren, das die Eingabe von Benutzer und Paßwort vorsieht, wie z.B. beim FTP-Protokoll, dort gibt es explizit die Befehle „USER <Benutzerkennung>“ und „PASS <Paßwort>“. Dadurch verwendet der Standard-Telnet-Client auch nicht die „~/.netrc“. Die von mir geschriebenen Telnet-Proxies füllen diese Lücke.

Für die Kommandozeile besteht der Telnet-Proxy aus einem Shell-Skript und dem eigentlichen Proxy. Das Shell-Skript sorgt dafür, daß der Proxy transparent für den Benutzer gestartet wird.

Der Aufruf erfolgt wie beim normalen Telnet mit

```
$ ssotelnet <myhost>
```

Das Shell-Skript startet dann den Proxy. Der Proxy sucht sich einen freien Port >1024 und meldet ihn dem Shell-Skript. Das Skript ruft dann „telnet“ mit den Parametern 127.0.0.1 und dem gemeldeten Port auf.

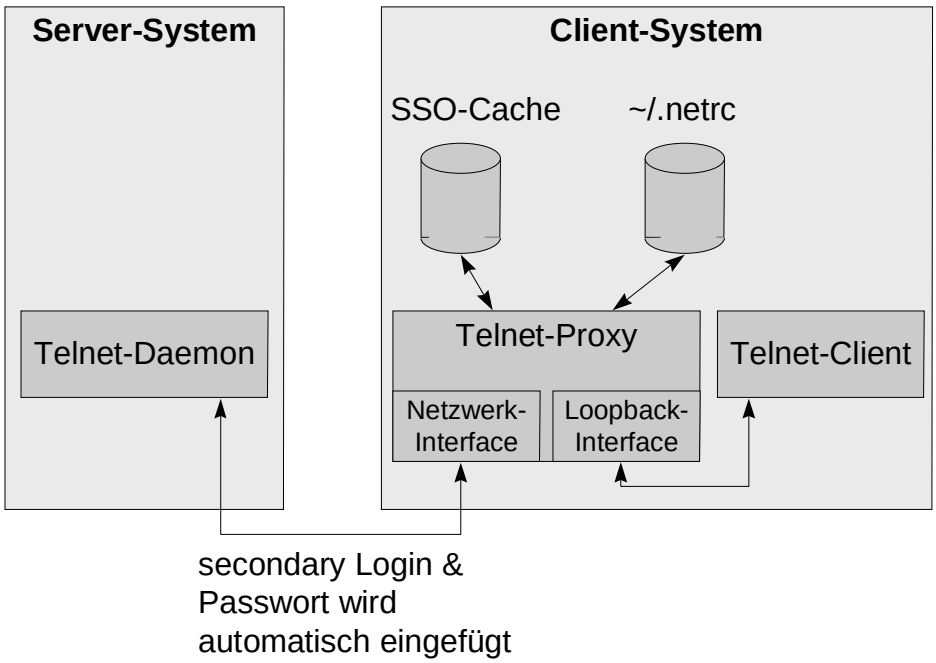
Nachdem der Proxy den ersten Connect erhalten hat, hört er auf, auf neue Clients zu warten und baut eine Verbindung zum Zielsystem auf. Der nun vom Server kommende Datenstrom wird in mehreren Stufen nach Mustern durchsucht.

Stufe	Muster	Aktion(en)
0	„ogin“ oder „nmelden“	Stufe = Stufe + 1
1	„: „	Secondary Login einfügen und Stufe = Stufe + 1
2	„assword“ oder „ennwort“	Stufe = Stufe + 1
3	„: „	Wenn „normaler“ Login : Secondary Password einfügen und Stufe = Stufe + 2 Wenn „loginiss“ Login : „pwd“ einfügen

		und Stufe = Stufe + 1
4	„>“	Secondary Password einfügen und Stufe = Stufe + 1
5	keins	Alles weiterleiten

Ab der Stufe 5 wird der Datenstrom in beide Richtungen nur noch weitergereicht.

Die secondary Paßwörter und Logins, die der Proxy benutzt kommen entweder aus AccessMaster oder aus der Datei ~/.netrc, wobei AccessMaster immer Vorrang hat. Systeme auf denen „loginiss“ installiert ist, stehen in /etc/amisshosts, dies ist deshalb notwendig, da man bei „loginiss“ als erstes Paßwort „pwd“ einfügen muß, damit man sich mit dem secondary Login anmelden kann. Normalerweise erwartet „loginiss“ das primary Login und Paßwort.



Konfiguration

/etc/amisshosts : Diese Datei enthält alle Hosts auf denen „loginiss“ installiert ist. Der Inhalt besteht aus den Namen, die durch einen Zeilenumbruch getrennt sind, z.B.

```
cips2
advm2
```

~/.netrc : Der Inhalt der netrc-Datei wird ebenfalls verwendet, wenn zu der Zielmaschine kein AccessMaster-Eintrag existiert. Das Format dieser Datei ist in der Man-Page auf dem jeweiligen System erklärt. Vom Proxy werden allerdings nur die Felder „machine“, „login“ und „password“ ausgewertet, daß sich das benötigte Format auf

```
machine <Name des Systems> login <Benutzerkennung> password <Passwort>
```

beschränkt.

Hier ein Beispiel :

```
machine cips2 login mylogin password mypassword
machine advm2 login mysecondlogin password mysecondpassword
```

Quelltext

Der Telnet-Proxy ist, abgesehen von einem Shellskript, in C geschrieben und sollte auf jedes Unix-Betriebssystem leicht portierbar sein. Die bisherigen Tests fanden auf AIX und Linux (ohne AccessMaster) statt.

Es gibt 7 Module. Das erste Modul, „amisshosts.c“, ist für das Lesen der Datei „/etc/amisshosts“ zuständig und enthält die Funktion „int isAmMachine(char* host)“. Diese Funktion liefert 1 wenn das System in der Datei enthalten ist und sonst 0.

Das nächste Modul ist „amuserpass.c“. Die einzige Funktion ist „int amuserpass(FILE* fd, char *system, char** user, char** password)“, sie enthält die Aufrufe an den SDM um für ein gegebenes System den Benutzernamen und das Passwort zu suchen. Das System muß dabei in der SIB im Feld „System:Identifier1“ stehen, welches dem „SSO Identifier 1“ in der

Systemmaske des „I.S.M. User Management Tool“ entspricht. Das Programm nutzt hier die selben Informationen wie „loginiss“ und der „Secured XDM“. Die Funktion gibt 0 zurück, falls der momentan angemeldete AccessMaster-Benutzer ein secondary Login auf dem Zielsystem hat, sonst 1. Der Filedeskriptor ist für eventuelle Fehlermeldungen gedacht.

Das dritte Modul ist „ruserpass.c“. Dieses Modul ist dem FTP-Client der University of Columbia, Berkley entnommen und nur leicht modifiziert um die Account- und Makro-Felder der Datei „~/.netrc“ zu ignorieren. Die Deklaration der Funktion ruserpass lautet wie folgt „int ruserpass(FILE* fd, char *host, char **aname, char **apass, char **aacct)“. Eingabeparameter sind der Filedeskriptor und der Systemname. Der Parameter „aacct“ wird zur Zeit nicht verwendet. Die Funktion gibt 0 zurück, wenn die Suche erfolgreich war und sonst -1. Außerdem enthält dieses Modul noch eine Hilfsfunktion, die den Inputstream aus der Datei in Tokens zerlegt.

Das Modul „daemon.c“ ist dem Buch „Unix Network Programming“ von W. Richard Stevens entnommen. Die Funktion „void daemon_start(int ignsigcld)“ dient im wesentlichen dazu, den Prozeß vom Terminal zu lösen. In der Funktion ist eine Änderung gemacht worden, um nur die Filedeskriptoren des Terminals zu schließen. Der Übergabeparameter ist für die Kontrolle eventueller Kindprozesse dieses Prozesses wichtig. Da keine Kindprozesse erzeugt werden, ist hier eine 0 als Parameter angemessen.

Das Modul „socket.c“ enthält eine Hilfsfunktion um mehrere Bytes in einen Stream Socket zu schreiben. Diese Funktion ist ebenfalls dem Buch „Unix Network Programming“ entnommen. Sie dient im wesentlichen als Ersatz für die Funktion „write(...)“ aus der Standard C-Library. Die Signatur ist „int writen(int fd, const char* ptr, int nbytes)“. Die Übergabeparameter sind der Filedeskriptor, das Zeichenarray und die Anzahl zu schreibender bzw. sendender Bytes.

Das Modul „util.c“ enthält drei Hilsfunktionen. Die erste „int strendcmp(const char *s1, const char *s2)“ vergleicht das Ende der Zeichenkette s1 mit der Zeichenkette s2. Sie gibt 0 zurück, wenn das Ende von s1 gleich s2 ist und sonst -1.
Die Funktion „void strcatchar(char *s, const char chr, int size)“ fügt der Zeichenkette s das Zeichen chr hinzu. Allerdings wird dabei eine maximale Länge von „size“ eingehalten. Ist die Zeichenkette länger, werden alle Zeichen um eine Stellen nach links verschoben und das erste Zeichen fällt weg.
Die letzte Funktion ist „void printchar(const unsigned char c)“. Sie wird bei der Debug-Ausgabe genutzt um Sonderzeichen durch ihre Escapesequenz darzustellen.

Das letzte Modul ist „tn-proxy.c“ und enthält das Hauptprogramm. Die Funktion „main“ versucht zuerst das secondary Login zu ermitteln. Als nächstes wird nachgesehen, ob auf diesem System „loginiss“ installiert ist. Nun sucht sich der Proxy einen Port > 1024, da er von einem Benutzer ohne Root-Rechte gestartet wird. Nachdem ein freier Port gefunden wurde, wird dieser auf der Standard-Ausgabe bekannt gegeben. Nachdem sich der erste Client verbunden hat, wird auf dem lokalen Port nicht weiter gehorcht und die Verbindung zum Remotesystem aufgebaut. Als nächster Schritt werden Login und Passwort eingefügt. Zum Schluß schaltet der Proxy in einen Transparenten Modus und schleust die Daten nur noch durch, ohne sie zu interpretieren.

Zusätzlich gibt es noch ein Shellskript, das den Proxy für den Benutzer transparent startet und dann den Telnet-Client mit dem gemeldeten Port startet.

```
#!/bin/sh

set `tn-proxy $1`
if [ $? -ne 0 ] ; then
      echo Error starting Daemon
      exit 1
fi
PORT=$3
echo Daemon startet at Port $PORT
sleep 1
```

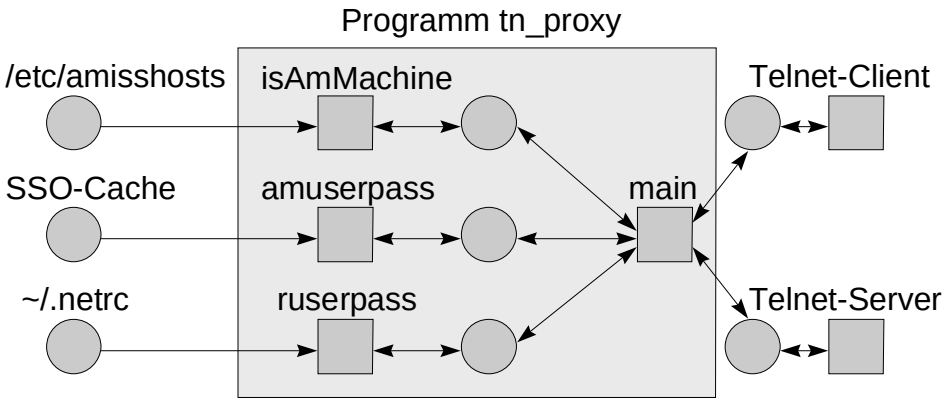
```
telnet 127.0.0.1 $PORT
```

Zuerst wird der Proxy gestartet und mit „set“ die Ausgabe ins Environment übernommen. Wenn der Proxy beim Starten einen Fehler hat, so endet er mit einem Errorlevel ungleich 0 und das Skript terminiert. Nach erfolgreichem Start des Proxies meldet dieser auf der Standard-Ausgabe

```
Socket : <Portnummer>
```

Die Portnummer steht dann in $3. Anschließend wird mit dieser Portnummer ein Telnet auf localhost gestartet.

Der Datenfluß stellte sich wie folgt dar :



Telnet-Proxy für Windows-Umgebungen

Funktion

Der Telnet-Proxy für Windows-Umgebungen funktioniert ähnlich wie der für die Kommandozeile. Der Unterschied besteht darin, daß er als ständig laufender Daemon konzipiert ist und standardmäßig auf Port 23 und IP-Adresse 127.0.0.1 auf Verbindungen wartet. Er ist multithreaded, so daß er mehrere Verbindungen auf einmal bearbeiten kann.

Wenn der Benutzer eine Telnet-Verbindung nach 127.0.0.1 aufbaut, öffnet der Proxy ein Fenster mit den bekannten Zielen. Der Benutzer muß nun ein Ziel

auswählen und den Connect-Button anklicken. Der Rest der Verbindung läuft dann wie beim Proxy für die Kommandozeile ab.

Der Proxy wird mit der Batch-Datei „start.bat“ gestartet.

Der Telnet-Proxy für Windows-Umgebungen ist in Java geschrieben und setzt deshalb eine Java-Runtime-Umgebung Version 1.1 vorraus.

Konfiguration

Unter Windows, die einzige Plattform die bisher getestet wurde, entspricht das Verzeichnis „/etc“ dem Verzeichnis „C:\ETC“. Die Unterscheidung wird automatisch durch das Programm vorgenommen.

Die Datei „/etc/amisshosts“ ist wie für den Telnet-Proxy für Kommandozeile, mit den Rechnern zu füllen, auf denen „loginiss“ installiert ist. Das Format ist gleich, ein Host pro Zeile.

```
advm2
cips2
```

Die Datei „/etc/tnproxyhosts“ enthält die Hosts, die in der Connect-Liste dargestellt werden. In einer Weiterentwicklung des Programms sollte diese Liste automatisch mit Hilfe der SSO-API und der Datei „~/.netrc“ erstellt werden. Das Format ist das selbe wie bei „/etc/amisshosts“.

```
advm2
cips2
advs2
```

Auch dieser Proxy wertet die Datei „~/.netrc“ aus. Unter Windows und anderen Betriebssystemen, bei denen Java kein Home-Directory finden kann heißt diese Datei „/etc/netrc“. Das Format ist wie unter Unix. Account und Macro-Einträge werden aus den bereits angegebenen Gründen ignoriert.

```
machine advs2 login ia140 password sag_ich_nicht
```

Quelltext

Der Hauptteil des Proxy ist in Java geschrieben, lediglich der Aufruf der AccessMaster-Funktionen ist durch eine DLL realisiert, die in C geschrieben ist.

Das Programm startet mit der Batch-Datei „start.bat“. Diese Datei enthält nur den folgenden Aufruf :

```
java Server
```

Der Aufruf kann evtl. noch um die Portnummer erweitert werden, auf der der Server horcht. Der Standard-Port ist, wie für Telnet üblich, 23.

Die Server-Architektur ist im wesentlichen dem Buch „Java in a Nutshell“ von David Flanagan entnommen.

Die Klasse Server, welche die main-Funktion enthält, ist abgeleitet von der Klasse Thread. Es gibt im wesentlichen den Konstruktor sowie die Funktionen „run()“ und „fail(...)“. Die Funktion „fail(...)“ gibt nur eine Fehlermeldung auf der Standardausgabe aus. Der Konstruktor initialisiert die statischen Variablen der Klasse und öffnet den Socket, allerdings horcht er nur auf 127.0.0.1, anschließend wird der Thread gestartet. Das Hauptfenster der Klasse MainFrame wird ebenfalls geöffnet. In der Funktion „run“ wartet der Server auf einen neuen Client und erzeugt für jeden Client eine Instanz der Klasse Connection.

Die statischen Variablen sind Instanzen der Klassen, die für Zugriffe auf den SDM und die Konfigurationsdateien zuständig sind. Diese Instanzen sind static und public, damit die Threads der einzelnen Verbindungen anschließend nicht jedesmal die Dateien neu öffnen und schließen müssen.

Die Klasse Connection besteht aus dem Konstruktor und der Funktion „run“. Für jede Verbindung, die durch den Proxy realisiert wird, ist eine Instanz der Klasse Connection verantwortlich. Die Klasse selbst ist ebenfalls von Thread

abgeleitet. Der Konstruktor übernimmt den Socket von der Klasse Server und öffnet den Dialog DialogSelectDestination, um das Zielsystem vom Benutzer zu erfragen. Danach sucht der Konstruktor zuerst in AccessMaster und dann in der netrc-Datei nach dem secondary Login und Paßwort. Nachdem die Verbindung zum Zielsystem aufgebaut ist, wird der Thread gestartet und der Rest der Arbeit wird von der Funktion „run“ übernommen. Die Funktion „run“ funktioniert nach dem gleichen Prinzip wie der „Telnet-Proxy für Kommandozeile“. Der Datenstrom wird in 5 verschiedenen Modi abgesucht. Im 5. Modus, nach dem Login, werden die Daten einfach nur noch hin und her gereicht. Die Funktion überprüft mit Hilfe der Klasse AccessMasterISSHosts ob auf dem Zielsystem loginiss installiert ist.

Die Klasse LastBuffer dient als Buffer für die letzten Zeichen. Dieser Buffer wird benutzt um das Ende des Datenstromes mit einer Zeichenkette zu vergleichen. Der Konstruktor erhält die Größe des zu erzeugenden Buffers. „addchar(...)“ fügt einen Buchstaben hinzu und mit „endswith(...)“ kann man das Ende des Buffers mit einer Zeichenkette vergleichen.

Das Hauptfenster wird von der Klasse MainFrame erzeugt. Es enthält ein Menu, mit dem man das Programm beenden kann und mit dem die Debug-Ausgabe im Hauptfenster ein- und ausgeschaltet wird.

Das Zielauswahlfenster wird von der Klasse DialogSelectDestination dargstellt. Diese füllt mit Hilfe der Klasse TelnetProxyHosts eine Drop-Down-Box mit den verfügbaren Hosts. Die DialogBox wird mit dem Connect-Button wieder geschlossen.

Der Konstruktor der Klasse TelnetProxyHosts liest die Datei „/etc/tnproxyhosts“ aus und legt die Namen in einem Vektor ab. Mit der Funktion „size(...)“ läßt sich die Anzahl der Verfügbaren Hosts feststellen. Die Hosts lassen sich mit „getHost(...)“ wieder auslesen.

AccessMasterISSHosts ist die Klasse, die die Datei „/etc/amisshosts“ ausliest. Das Auslesen der Datei geschieht durch den Konstruktor. Ob ein Hostname in der Datei vorkommt, kann man mit der Funktion „isISSLoginInstalled(...)“ überprüfen.

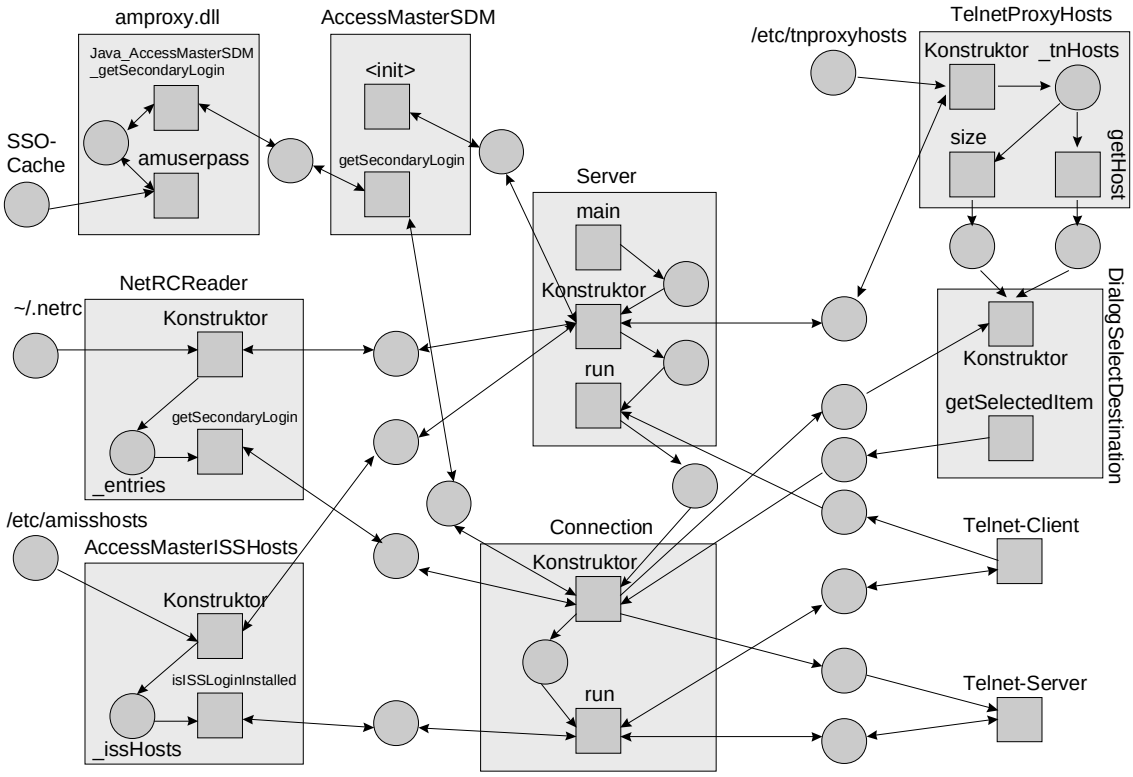
Die beiden Klassen, AccessMasterSDM und NetRCReader, die das secondary Login liefern, benutzen beide die Klasse SecondaryLogin als Return-Wert. Die Klasse hat einen Default-Konstruktor, der ein leeres secondary Login erzeugt und einen Konstruktor, der die beiden Membervariablen direkt mit dem Werten der Übergabeparameter belegt. Außerdem gibt es noch für beide Membervariablen entsprechende get- und set-Funktionen, um die Variablen zu lesen, bzw. zu setzen.

Die Klasse NetRCReader liest die netrc-Datei ein. Um die Einträge der Datei in einer Hashtabelle zu verwalten, enthält die Klasse die Unterklasse MachineEntry. Der Konstruktor der Klasse NetRCReader liest die netrc-Datei ein und legt die Einträge in einer Hashtabelle ab. Der Hashwert ist dabei der Hostname. Die Funktion „getSecondaryLogin(...)“ liefert den secondary Login für die angegebene Maschine. Die Unterklasse Token dient als Hilfsklasse für das Parsen der netrc-Datei. Die Funktion „getToken(..)“ der Klasse NetRCReader liefert jeweils das nächste Token des übergebenen Eingabestreams.

Die Klasse AccessMasterSDM hat nur eine native Funktion „getSecondaryLogin(...)“ und den statischen Aufruf zum Laden der DLL.

Mit dem Java-Tool „javah“ ist die Header-Datei „AccessMasterSDM.h“ erzeugt worden. Die Funktion „Java_AccessMasterSDM_getSecondaryLogin(...)“ ist in der Datei amproxy.c mit Quelltext gefüllt. Die Funktion ruft die Funktion „amuserpass(...)“ auf. „amuserpass(...)“ entspricht der gleichnamigen Funktion beim „Telnet-Proxy für Kommandozeile“.

Die Beziehungen zwischen den einzelnen Klassen sehen wie folgt aus (nur die Klassen und Module die für das Verständnis wichtig sind) :



FTP-Proxy für Verbindungen von Nicht-Unix-Systemen

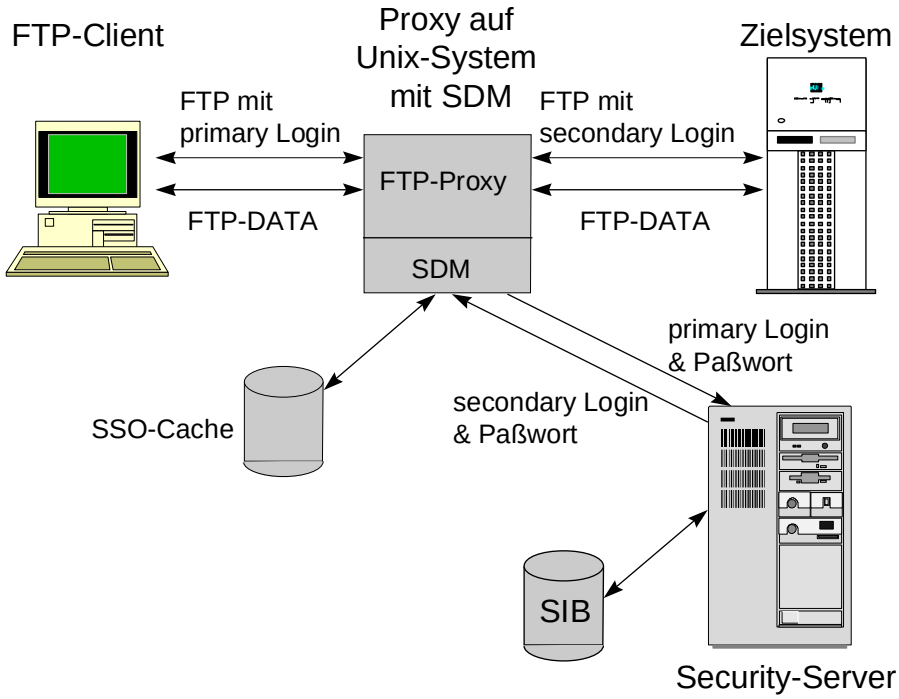
Funktion

Der FTP-Proxy wird auf einem der Unix-Systeme mit SDM installiert und dient Windows Workstations und Workstations ohne SDM als Proxy, um auf die verwalteten Unix-Server mit FTP zuzugreifen.

Der Login erfolgt mit „<primary Login>@<Zielsystem>“, wobei Zielsystem der „SSO Identifier 1“ eines Systems in der SIB sein muß. Der Proxy baut dann eine FTP-Verbindung zum Zielsystem auf und meldet den Benutzer dort mit secondary Login und Paßwort an.

Bei der folgenden Session werden auf der Control-Session alle Befehle, bis auf „PORT“, „USER“, „PASS“ und „PASV“, transparent durchgeschleust. Das

PORT-Kommando wird abgefangen und für den Aufbau der Datensession zum Client benutzt. Für das Zielsystem wird entsprechend ein neues PORT-Kommando generiert. Das USER- und das PASS-Kommando werden ignoriert und mit einer positiven Antwort an den Client quittiert, da der Benutzer bereits durch den Proxy angemeldet wird und die secondary Logins und Paßwörter nicht kennt. Das PASV-Kommando ist noch nicht implementiert und wird mit einer entsprechenden Fehlermeldung quittiert.



Der Proxy wird mit „ftp-proxy <Portnummer>“ gestartet, wobei die Angabe der Portnummer optional ist. Wird die Portnummer nicht angegeben, so wird standardmäßig 21, der Standard FTP-Port, benutzt.

Konfiguration

Eine Konfiguration ist nicht erforderlich.

Quelltext

Der FTP-Proxy basiert vom Prinzip auf dem FTP-Proxy aus dem Firewall-Toolkit der Firma Trusted Information Systems. Allerdings handelt es sich

hierbei um eine weitgehende Neuentwicklung. Es wurden nur Hilfsfunktionen wie „say(...)“, „sayn(...)“, „timeread(...)“ und „porttoaddr(...)“ übernommen. Insbesondere das Session-Handling wurde neu entwickelt.

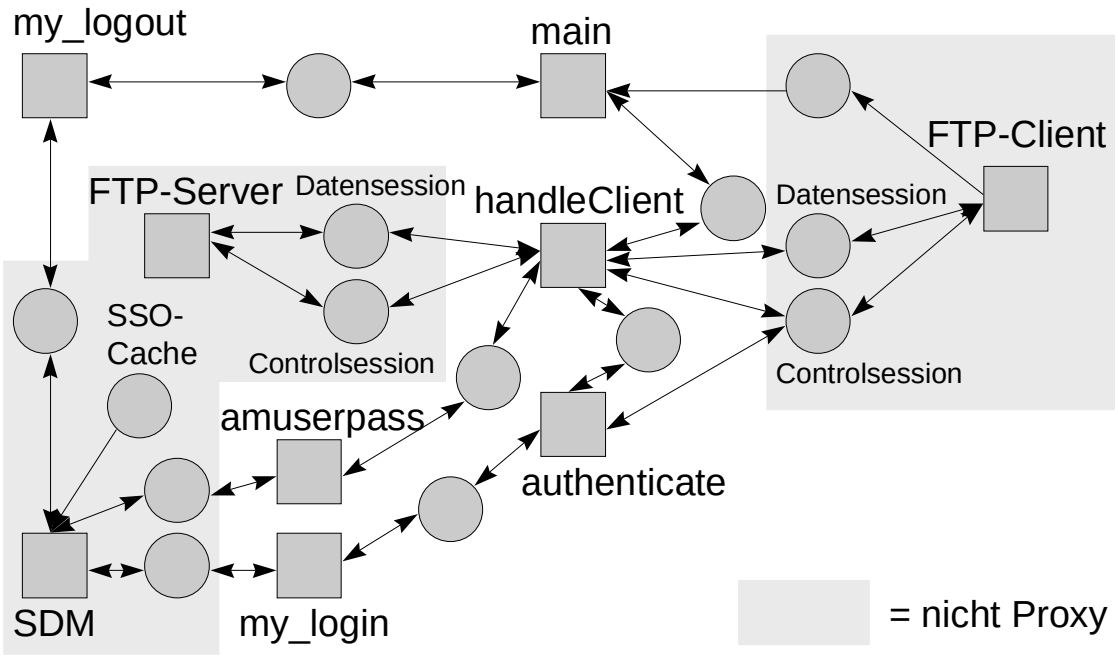
Das Modul „ftp-proxy.c“ enthält einige Hilfsfunktionen und das Hauptprogramm. Die Funktion „int main(int argc, char* argv[])“ öffnet den Socket, standardmäßig 21, und wartet auf Verbindungen. Sobald ein Client die Verbindung zum Proxy aufgebaut hat, forkt sich der Prozeß und übergibt die Verbindung im Kindprozeß an die Funktion „int handleClient(int local_fd)“.
In dieser Funktion befindet sich die eigentliche Arbeitsschleife. Die Funktion führt zuerst eine Authentisierung des Client gegen AccessMaster mit Hilfe der Funktion „int authenticate(int fd, char* remote_host)“ durch. Mit den von AccessMaster erhaltenen Daten wird der Benutzer dann auf dem Zielsystem angemeldet. Nachdem der Benutzer am Zielsystem angemeldet ist geht die Funktion in die Proxy-I/O-Schleife. In dieser Schleife werden die Befehle der Control-Session fast transparent durchgereicht. Das USER- und das PASS-Kommando wird nicht an das Zielsystem weitergereicht und mit einer positiven Antwort an den Client beantwortet, da der Benutzer vom Proxy auf dem Zielsystem mit dem secondary Login angemeldet wird und der Benutzer das secondary Login und Paßwort nicht kennt. Das PASV-Kommando wird mit einer negativen Antwort vom Proxy beantwortet, die besagt, daß dieses Kommando noch nicht implementiert ist. Für das Handling der Datensessions wird das PORT-Kommando von der Funktion „int command_port(int local_fd, int remote_fd, char *param, int* local_data_fd, int* remote_data_sockfd)“ verarbeitet.
Die Funktion „command_port(...)“ sorgt dafür, daß die Parameter des PORT-Kommandos mit Hilfe der Funktion „int porttoaddr(char *param, struct sockaddr_in* retSock)“ dekodiert werden und die Verbindung zum Client aufgebaut wird. Anschließend wird ein Socket für die Datenverbindung vom Server zum Proxy erzeugt und die Parameter mit einem PORT-Kommando an den Server geschickt. Sobald der Server das Kommando positiv beantwortet hat, wird eine positive Antwort an den Client geschickt und die Funktion hat

ihre Aufgabe erfüllt.
Ansonsten enthält das Modul „ftp-proxy.c“ noch einige Hilfsfunktionen. Die Funktion „int say(int fd, char* s)“ schreibt mit Hilfe von „int sayn(int fd, char* s, int n)“ einen String in den übergebenen Deskriptor. Die Funktion „int sayn(int fd, char* s, int n)“ schreibt die übergebene Zeichenkette der Länge „n“ in den Deskriptor und hängt einen Zeilenumbruch an. Die beiden Funktionen werden benutzt um Befehle zum Server oder Antworten zum Client zu schicken. Die Funktion „int timegetline(int fd, unsigned char* buf, int size)“ ließt eine Zeile vom übergebenen Deskriptor unter Berücksichtigung des globalen Timeouts „PROXY_TIMEOUT“. Um einzelne Zeichen zu lesen wird die Funktion „int timeread(int fd, char* buf, int size)“ benuzt. Für das Lesen von Zeilen aus Deskriptoren, die das Attribut „O_NONBLOCK“ haben, wird die Funktion „int getline(int fd, unsigned char* buf, int size, int* fdstat)“ benutzt.

Das Modul „amutils.c“ enthält die schon vom POP3-Daemon bekannten Funktionen „my_login(...)“ und „my_logout()“ mit denen das Login und Logout bei AccessMaster durchgeführt wird.

Die Module „amuserpass.c“, „socket.c“ und „daemon.c“ sind identisch mit denen beim Telnet-Proxy.

Die Kommunikation der wesentlichen Funktionen des FTP-Proxy stellt sich wie folgt dar :



Architektur

Nachdem die Einführung von AccessMaster abgeschlossen ist und das Single Sign-On aktiviert ist, stellt sich die folgende Architektur dar.

Telnet

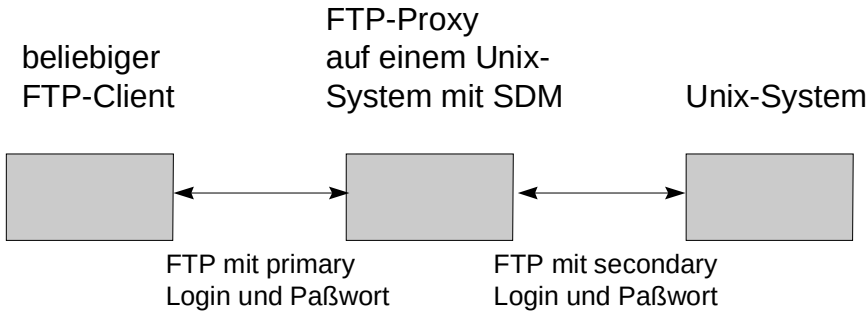
Damit Telnet von überall her mit dem primary Login möglich ist, wird auf allen verwalteten Unix-Servern das Modul „loginiss“ und der AccessMaster „Secured XDM“ aktiviert. Damit die Benutzer, die sich auf den Unix-Servern mit dem primary Login authentisiert haben, mit dem dann dort vorhandenen SSO-Cache auf einen weiteren Unix-Server wechseln können, wird auf den Servern der Telnet-Proxy für Kommandozeile installiert.

Auf den Windows-Workstations, auf denen der AccessMaster-Login installiert ist, wird der Telnet-Proxy für Windows-Umgebungen installiert. Dadurch können die Benutzer der Workstations ohne Eingabe des Paßworts mit Telnet auf den Unix-Server arbeiten. Das secondary Login und Paßwort wird automatisch, unter Verwendung der SSO-API, in den Datenstrom eingestellt.

FTP

Um FTP von einem Unix-Server zum anderen zu machen, muß jeder Benutzer das Programm „create_netrc“ in sein Login-Skript eintragen. Während des FTP-Logins zu einem anderen Unix-System nimmt der FTP-Client die Daten aus der Datei „~/.netrc“.

Von allen anderen Systemen aus geht jeder FTP über den FTP-Proxy.



Der FTP erfolgt mit Ziel Proxy und Login „<primary_Login>@<Zielsystem>“, der Proxy stellt dann die Verbindung zum Zielsystem mit dem richtigen secondary Login und Paßwort her.

POP3

Für POP3 wird auf allen Unix-Servern der vorhandene POP3-Daemon durch den POP3-Daemon mit AccessMaster-Authentisierung ersetzt. Die Abholung der Mail kann dann wahlweise mit dem primary oder dem secondary Login und Paßwort erfolgen, wobei das primary Login bevorzugt wird.

Außerdem sei hier nochmals explizit darauf hinweisen, daß bei Verwendung von secondary Logins die Mengen der primary und secondary Logins disjunkt sein müssen.

Sonstige

Der Netzwerkzugang über die Ascend wird mit dem Radius-Dienst von AccessMaster auf dem Security-Server realisiert. Die Authentisierung erfolgt mit primary Login und Paßwort.

Für den PC-Pool werden die Benutzer mit AccessMaster verwaltet. Das NT-Login von Microsoft wird durch das von AccessMaster ersetzt, wodurch der Zugriff auf die Netzressourcen transparent ist.

NIS wird offiziell leider nicht unterstützt. Allerdings haben Tests ergeben, daß man Benutzer und Gruppen unter AIX anlegen kann, die nachher mit „make all“ im Verzeichnis „/var/yp“ in der NIS-Domain bekannt gemacht werden. Diese Aufgabe kann ein cron-Job übernehmen.

Einführungsphase

Dieser Abschnitt beschreibt die Vorgehensweise für die Einführung von AccessMaster im ADV-Labor, sowie besondere Einstellungen, die nicht in der vorhandenen OpenMaster-Dokumentation enthalten sind.

Installation und Konfiguration

Security Server

An erster Stelle der Einführung steht die Installation und Konfiguration des Security Servers.

Oracle

Oracle Version 7.3.2 oder 7.3.3 wird nach dem „Oracle Installation Guide“ von OpenMaster installiert.

AccessMaster

Die Installation des Security Servers wird nach dem aktuellen SRB und TSB des ISM Support Teams durchgeführt. Als Installationsverzeichnis, auf welches später mit „$ISMROOT“ Bezug genommen wird, sollte /ism454 gewählt werden.

Für die vorhandene Umgebung sollten folgende Komponenten installiert werden :

Von der ersten CD

- ISM_BASE
  (Framework und Basisdienste von OpenMaster)
- ACCESS_MASTER
  (AccessMaster selbst, sowie Unix User Management und RADIUS/TACACS+ Server)

- SNMP_AI
  (SNMP Agent Integrator)
- WGMNG_NT
  (Workgroup Management und User Management für Windows NT)

Von der Agent-CD

- AC_MASTER_AGT
  (AccessMaster Agenten, die die User Management Funktionen zur Verfügung stellen)
- ISM_AGENTS
  (generelle OpenMaster Agenten, wie der SNMP Daemon Dispatcher)

Falls Entwicklungen mit dem Generic Agent oder den APIs gemacht werden sollen, werden von der Toolkit-CD folgende Pakete benötigt

- LIB_TOOLKIT
  (OpenMaster Toolkit)
- AM_SNMP_TLK
  (AccessMaster Toolkit)
- AM_BSS-GSS_API
  (AccessMaster Client API)

Um die Installation abzuschließen sind noch die aktuellen Fixes von CD oder Tape einzuspielen.

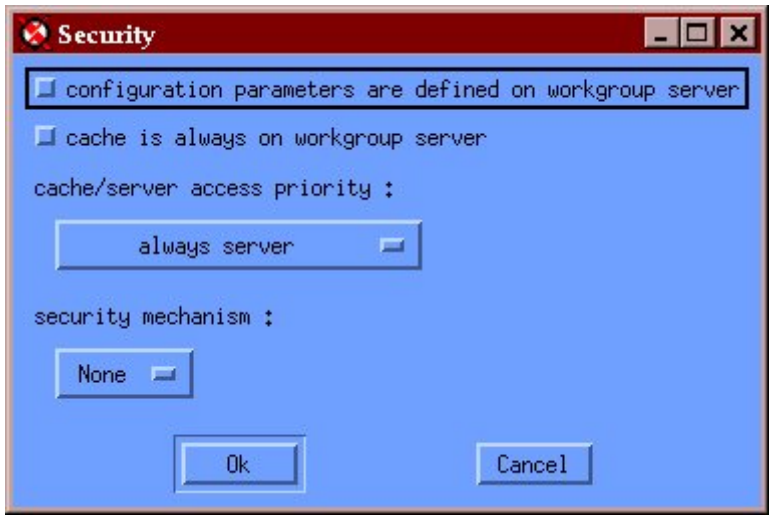
Damit ist die erforderliche Software soweit vorhanden. Die Konfiguration erfolgt jetzt nach dem TSB, allerdings sollte vor der Generierung der SIB mit „init_db“ der Parameter „UnixUserIdentifier“ aus der Datei „$ISMROOT/var/IUM_UM/IUMConfig“ auf einen Wert gesetzt werden, der größer ist, als die größte auf den Unix-Systemen vergebene UserId. Der Grund hierfür ist der, daß AccessMaster jedem Benutzer eine Default-UserId vergibt. Ist diese Id auf dem Zielsystem aber schon vergeben, so gibt es bei der automatischen Registrierung Probleme und man muß den Wert jeweils vorgeben, bzw. auf „0“ setzten, damit er vom Zielsystem beliebig gewählt wird.

Um die Benutzer auf dem Security Server selbst verwalten zu können, müssen die entsprechenden Agenten lokal installiert werden. Hierbei ist darauf zu achten, daß die Agenten zwingend unter „$ISMROOT“ installiert werden müssen. Die benötigten Agenten sind :

- AccessMaster_file_transfert_server
- AccessMaster_Security_Data_Manager
- AccessMaster_workgroup_file_server
- Integrated_User_Management_Agent
- SNMP_Dispatcher_Daemon

Der „Security Data Manager“, kurz SDM, wird insbesondere für Tests mit dem Tool „iss_tty“ und den Radius Server benötigt.

Der SDM muß anschließend mit dem Tool „rc.issconf“ installiert werden. Dabei ist die Cache-Konfiguration wie folgt vorzunehmen.



Die Einstellung „always Server“ bewirkt, das der SSO-Cache auf dem Security Server nie genutzt wird und die Authentisierung immer direkt gegen AccessMaster geschieht. Dies ist nur solange sinnvoll, wie man den SDM auf dem Security Server für Tests benutzt. Wird auf dem Security Server auch das Modul „loginiss“, der „Secured XDM“ oder der RADIUS-Server gestartet, sollte diese Einstellung auf „first Server, then Cache“ geändert werden, damit bei

einem Ausfall der Authentisierungs-Prozesse eine Authentisierung gegen den Cache möglich ist.

Die Workgroup des Security Servers ist zwingend „AdministrationCenter“.

Die Installation der Agenten erfolgt wie im TSB beschrieben.

Radius Server

Für den Radius-Server wird als erstes die Dictionary-Datei der Ascend benötigt. Diese Datei enthält ein Mapping zwischen IDs, logischen Namen und Datentypen. Die Datei muß nach „$ISMROOT/var/config/AM_NAS/radius.dictionary“ kopiert werden und um die AccessMaster-spezifische ID Role wieder ergänzt werden.

```
ATTRIBUTE   Role                        224  string
```

Für das Mapping zwischen Parametern aus der SIB und RADIUS-Feldern ist einmal ein Service RADIUS in der SIB anzulegen und die Beziehungen zwischen SIB-Parametern und RADIUS-Parametern müssen in der Datei „$ISMROOT/var/config/AM_NAS/raduis.sso“ angegeben werden.

Ich habe bisher keine Tests mit der Ascend unternommen, da während dieser Zeit keine Benutzer, außer den AccessMaster-Benutzern, den Einwähldienst nutzen können. Deshalb ist im folgenden die „radius.sso“, die mit einer Bintec Bianca/Brick XS und dem RADIUS-Client von Merit getestet worden ist, abgedruckt. In diesem Fall wird der Benutzer einfach nur authentisiert und jeder bekommt die Attribute „Framed-User“, „PPP“ und Timeout 3600.

```
[default]
# Service-Type = Framed-User
6="2"
# Framed-Protocol = PPP
7="1"
# Session-Timeout = 3600
27="3600"
# Idle-Timeout = 3600
28="3600"

# The priority entry descriptions will override the default
# descriptions if any overlap (i.e. : a field is described in both
# entries). See the default entry for the syntax.
```

```
[priority]

#Vendor specific Things

[166]
#1="/d1",
#2="\x01", Service:Cust:Descript

[200]
#10="TEST"
```

Zu guter letzt müssen noch die Paßwörter für die RADIUS-Kommunikation mit dem Tool „config_amnas“ angegeben werden.

Weitere Hinweise und Anleitungen für getestete Router stehen im „AccessMaster Configuration Guide“.

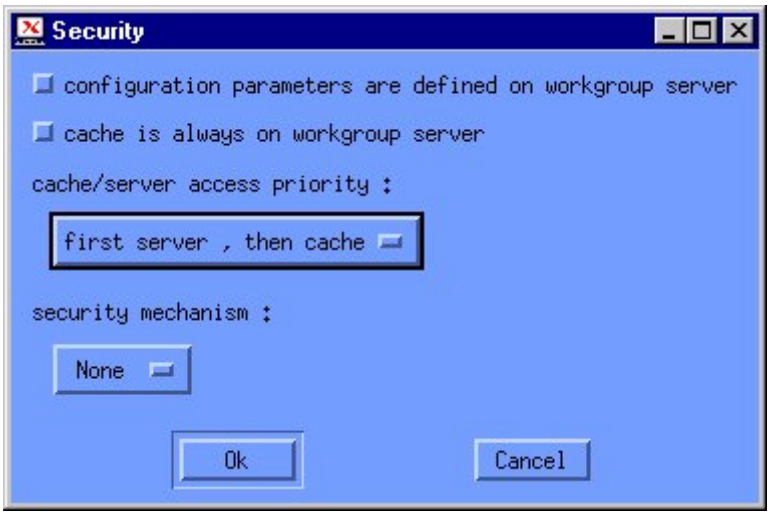
Unix Workgroup Server

Für die Unix Workgroup Server ist ein Agentenpaket ähnlich dem des Security Server zu erstellen.

- AccessMaster_Secured_XDM_End_Login
- AccessMaster_file_transfert_server
- AccessMaster_Security_Data_Manager
- AccessMaster_workgroup_file_server
- Integrated_User_Management_Agent
- SNMP_Dispatcher_Daemon

Das Zielverzeichnis sollte mit „/usr/ismAgents“ für alle Unix Workgroup Server gleich gewählt werden.

Der „Secured XDM“ und der „Security Data Manager“ ermöglichen es den Benutzern sich mit dem primary Login und Paßwort anzumelden. Die Einstellung der Cache-Parameter des SDM sollte wie folgt aussehen :



Die SNMP-Agenten („Integrated User Management Agent“ und „SNMP Dispatcher Daemon“) sind für die Verwaltung der Zugangsberechtigungen auf dem Server. Die Konfiguration der Agenten erfolgt nach dem TSB.

Der „File Transfer Server“ und der „Workgroup File Server“ werden benötigt, wenn aus dem Server ein Workgroup Server im Sinne von AccessMaster (siehe Glossar) werden soll. Dies kann nötig werden, falls die direkte Übertragung der Audit-Daten zu viel Netzwerk-Traffic erzeugt, das kann allerdings erst der laufende Betrieb zeigen.

Auf jedem Unix Server wird außerdem der „Telnet-Proxy für Kommandozeile“, das „Programm zur Generierung von „~/.netrc““ und der „POP3-Daemon mit AccessMaster-Authentisierung“ installiert. Auf einem der Unix-Server wird zusätzlich noch der „FTP-Proxy für Verbindungen von Nicht-Unix-Systemen“ installiert.

Windows NT Workgroup Server

Für den Windows Workgroup Server kann man entweder Disketten mit dem Tool „$ISMROOT/remote/AccessMaster/InstallDisk/creatdsk“ erstellen oder aber die Dateien per FTP auf den Windows NT Server kopieren.

Benötigt werden, in Klammern Position im Dateisystem :

- Windows NT Workgroup Server
  ($ISMROOT/remote/AccessMaster/wkserver/nt)
- ISM-UM Windows NT Agent
  ($ISMROOT/remote/AccessMaster/iumagent/nt)
- Basic Security Service for Windows NT
  ($ISMROOT/remote/AccessMaster/dc_wkst/win32)

Der „ISM-UM Windows NT Agent“ wird abweichend installiert, so daß er direkt Requests vom Security Server akzeptiert. Im Framework ist dann entsprechend der Port 3764 (IUM-Agent) anstatt 161 (Standard-SNMP) für den NT-Server anzugeben.

Der SDM (ist im Basic Security Service for Windows NT) wird auch auf dem Windows NT Server benötigt und wird wie auf dem Unix Workgroup Server konfiguriert. Bei der Installation ist es, mindestens unter Windows NT 4.0, erforderlich die Sprachumgebung während der Installation auf Englisch zu stellen, da das Installationsprogramm mit den deutschen Ländereinstellungen Probleme hat.

Um SSO-Telnet zu ermöglichen, wird auf dem NT Server der „Telnet-Proxy für Windows-Umgebungen“ und das Java Runtime Environment 1.1.x benötigt.

Windows NT Workstation

Auf den Windows NT Workstations wird lediglich das Paket „Basic Security Service for Windows NT“ benötigt. Die Installation und Konfiguration erfolgt analog zum Windows NT Server. Für den SSO-Telnet ist hier wie beim NT

Server der „Telnet-Proxy für Windows-Umgebungen“ und das Java Runtime Environment 1.1.x erforderlich.

Aufbau der Organisation in AccessMaster

Die Organisation wird, wie bereits erwähnt, nach X.521 in AccessMaster aufgebaut.

Für den Aufbau der Organisation ist die Struktur von Seite 26 vorgesehen. Demnach sind 2 „User Roles“ und eine „Time Table“ anzulegen.

Die User Roles UR-Student und UR-Labormitarbeiter sind mit den Default-Werten angelegt. Sie existieren momentan nur deshalb, weil jeder Benutzer eine Rolle haben muß. Die Unterscheidung zwischen Studenten und Mitarbeitern ist trotzdem vorsorglich getroffen worden, falls in Zukunft DCE- oder Sesame-Anwendungen zum Einsatz kommen, oder Desktops zugewiesen werden sollen.

Die Time Table TT-ALLDAYLONG ist für alle vorgesehen, damit man sich zu jeder Zeit anmelden kann. Sie legt fest, daß sich diejenigen, denen diese Zeittafel zugewiesen ist, jeden Tag zu jeder Zeit anmelden können.

Die Benutzerdaten eines Studenten sehen wie folgt aus :
(Felder die nicht geändert werden können sind nicht enthalten)

Feld	Inhalt
ISM-UM Name	IA300
Name	Kriener
First Name	Thomas
Identification Code	10232135
Audit Identifier	IA300, Thomas Kriener
Expiry Date	MM/DD/YYYY
Welcome Language	en
Imposed Profile	

User Role	UR-STUDENT
Privileges	
Groups of Privileges	
Audit	Yes
Sessions Configuration	Standard Session Configuration
Password Policy	Administrator Password Policy
Primary Password	----------
Change Password At Next Login	Yes
Access Locked	No
User Message	
Unix Default Login Name	ia300
User ID (Unix/Dce; 0='chosen by target')	5004
GCOS7 Default Login Name	IA300
GCOS8 Default Login Name	IA300
DCE Default Login Name	IA300
NetWare Default Login Name	IA300
Lan Manager Default Login Name	IA300
Windows NT Default Login Name	IA300
RACF Default Login Name (User-Id)	IA300
TopSecret Default Login Name (ACID)	IA300
Generic NT Default Login Name	IA300
TopSecret Default Department (ACID)	
CA-ACF2 Default Login Name (LID)	
Without Target Default Login Name	IA300
Phone	
Postal Address	Ohler Str. 5\n51766 Engelskirchen
Organization Name	
Organization Number	
EMailAddress	T.Kriener@bull.de

Für einen Mitarbeiter ändert sich im wesentlichen nur das Feld „User Role“ in UR-LABORMITARBEITER.

Generierung der AccessMaster Benutzer

Die Benutzer liegen in einer ASCII-Datei vor und werden mit entsprechenden Import-Filtern in AccessMaster importiert.

Die ASCII-Datei enthält folgende Daten

- Benutzerkennung (z.B. AI300)
- Name (z.B. Kriener)
- Vorname (z.B. Thomas)
- Paßwort (z.B. kriener35)
- Matrikelnummer (z.B. 10232135)
- EMail-Adresse (z.B. T.Kriener@bull.de)
- Adresse (z.B. Ohler Str. 5\n51766 Engelskirchen)

Die Benutzerkennung ist wie bisher zwei bis drei Buchstaben und eine laufende Nummer. Die Nummer sollte so hoch gewählt werden, daß sie bisher noch auf keinem System vorhanden ist, dadurch ist beim Anlegen einer neuen Kennung gewährleistet, daß diese auf dem Zielsystem noch nicht existiert.

Die Buchstaben identifizieren den Studiengang und werden von Herrn Gesper verwaltet. Bisher sind unter anderem folgende Kürzel festgelegt :

Studiengang	Kürzel
Allgemeine Informatik	AI
Technische Informatik	TI
Wirtschaftsinformatik	WI
Wirtschaftszusatzstudium	WZ
Verbundstudiengang	VB
Maschinentechnik	MM
Wirtschaftsingenieur	MW

Elektrotechnik	EI
Sonstige	SS
Lehrer von „Schulen ans Netz“	LE
Schüler von „Schulen ans Netz“	SCH

Name und Vorname sind wie üblich formatiert.

Als Default-Passwort ist der Nachnamen plus die letzten zwei Ziffern der Matrikelnummer vorgesehen. Um das bisherige Format ($?<*Nachname*>) beizubehalten müßte die Paßwort-Policy angepaßt werden.

Eine Import Datei hat folgendes Format :

```
## Import persons

@@ Name                  ;;
   RealName              ;;
   FirstName             ;;
   AuditIdentifier       ;;
   OrganismNumber        ;;
   ExpiryDate            ;;
   Parent                ;;
   UserRole              ;;
   WorkGroups            ;;
   TimeTables            ;;
   Audit                 ;;
   AuthenticationMethod  ;;
   Pass                  ;;
   PassChangeAtNextLogin ;;
   UNIXID                ;;
   PostalAddress         ;;
   EMailAddress          ;;

@@ AI300 ;; Kriener ;; Thomas ;; AI300, Thomas Kriener ;;
   10232135 ;; ;;
   #Top/DE/FH-Koeln/GM/FB Informatik ;;
   #Top/DE/FH-Koeln/UR-STUDENT ;;
   #Top/.IUM Organization/* ;;
   #Top/DE/FH-Koeln/TT-ALLDAYLONG ;;
   True ;; Password ;; Kriener35 ;; True ;; ;;
   Ohler Str. 5\n51766 Engelskirchen ;;
   T.Kriener@bull.de ;;

@@ AI301 ;; Linke ;; Mario ;; AI301, Mario Linke ;;
   LINKE ;; ;;
   #Top/DE/FH-Koeln/GM/FB Informatik ;;
   #Top/DE/FH-Koeln/UR-STUDENT ;;
   #Top/.IUM Organization/* ;;
   #Top/DE/FH-Koeln/TT-ALLDAYLONG ;;
   True ;; Password ;; Linke07 ;; True ;; ;;
```

```
Am Sandberg 1\nGummersbach ;;
linke@gm.fh-koeln.de ;;
```

In diesem Beispiel werden zwei Studenten importiert. Wie man leicht erkennen kann, gibt es einen Kopf, der das Format der folgenden Datensätze angibt. Die Felder werden durch „;;“ getrennt, Datensätze beginnen mit „@@“. Da davon auszugehen ist, daß die Verwaltung ein anderes Format zur Verfügung stellt, wird es erforderlich sein, ein Konvertierungsprogramm zu schreiben. Da es sich hier um einfache Plain-ASCII-Dateien handelt, ist der Aufwand minimal.

Import der bestehenden Benutzerkennungen

Vor dem Import bestehender Benutzerkennungen von der ADVM2 und CIPS2 müssen zuerst die Services (Unix-Gruppen) importiert werden. Dafür reicht es allerdings die Services zu importieren, die von „normalen“ Benutzern verwendet werden. Mit „normalen“ Benutzern sind alle Benutzer außer Systembenutzern, wie root, adm, mail, daemon usw., gemeint.

Achtung : Einmal importierte Services und Registrierungen lassen sich normalerweise nicht aus AccessMaster entfernen, ohne daß sie auf dem Zielsystem gelöscht werden.

Im nächsten Schritt können nun die Registrierungen importiert werden. Ein Problem das hierbei auftreten wird, ist daß die Kennung auf dem Zielsystem nicht der AccessMaster-Kennung entspricht. Meine Empfehlung ist es eine Liste nach Namen sortiert, mit folgenden Feldern zu erstellen.

- Name
- Vorname
- AccessMaster-Kennung

Beim Import kann man dann in einem Telnet-Fenster mit

```
finger <Benutzername>@<Systemname>
```

herausfinden, welchen Namen der Benutzer hat und in der Liste die Kennung nachschlagen. Dieses Verfahren ist nicht das effizienteste, weil es sich hier um einen einmaligen Vorgang handelt, ist es aber vertretbar.

Die Erstellung eines Programms mit Hilfe der Administration-API ist zwar denkbar, hat aber einige Tücken, wenn ein Benutzer das Bemerkungsfeld in der /etc/passwd geändert hat. Außerdem bewerte ich den Programmieraufwand inklusive Fehlerbereinigung höher als den des manuellen Imports.

Beim Import bestehender Registrierungen wird AccessMaster verlangen, daß falls der Benutzer vorher ein Paßwort hatte ein neues Paßwort vergeben wird, damit AccessMaster das Paßwort kennt. Da der Import von ca. 600 Registrierungen pro System länger als einen Tag dauern wird, bekommt der Benutzer die Möglichkeit, das Kennwort wieder auf den alten Wert zu setzen, was dann zur Folge hat, das AccessMaster zwar die Registrierung kennt, nicht aber das Paßwort. Sobald nun Single Sign-On aktiviert wird, muß das Paßwort auf dem Zielsystem neu gesetzt werden. Dies kann mit einem Programm erfolgen, daß von einem norwegischen Kollegen erstellt wurde. Diesem Programm wird eine Liste übergeben, deren Zeilen folgendes Format haben :

```
<System>     <secondary Login>     <secondary Paßwort>
```

Das Programm setzt dann entsprechend das secondary Paßwort in der SIB und auf dem Zielsystem neu, so daß die Informationen in der SIB und auf dem Zielsystem wieder synchron sind. Bei Bedarf kann das Programm auch zufällige Paßwörter vergeben.

Einführung von Single-Sign On

Die Einführung der SSO-Funktionalität kann System für System erfolgen, wobei alle Benutzer des jeweiligen Systems unter der Verwaltung von AccessMaster sein sollten, bevor „loginiss“ und der „AccessMaster Secured XDM“ installiert werden. Von nun an können sich alle Benutzer mit dem primary Login authentisieren.

Damit der Komfort des SSO von Anfang an richtig zum Tragen kommen kann, sollte mit den Windows-Systemen begonnen werden.

Spätestens nach dem Import der Benutzer der ADVM2 in die SIB und der Aktivierung des SSO muß der Radius-Dienst von der ADVM2 auf den AccessMaster Server umgestellt werden, da die Benutzer ihr Paßwort auf der ADVM2 nicht mehr kennen.

Ich empfehle also folgende Reihenfolge

1. Windows-NT Server importieren
2. SDM und Telnet-Proxies auf den Windows-Systemen installieren
3. Radius-Server auf AccessMaster umstellen
4. Unix-Systeme nacheinander importieren und mit „AccessMaster Secured XDM“ und „loginiss“ versehen.

Sicherheitsgewinn durch den Einsatz von AccessMaster

AccessMaster kann die Sicherheit innerhalb des verwalteten Bereichs erheblich verbessern. Den Benutzern wird ein für Ihre Gruppe eindeutiges Profil gegeben, das die Übersicht erleichtert.

Die Administration kann unabhängig von den Systemadministratoren sicher delegiert werden. Ein Datenbankadministrator kann das Recht erhalten, Datenbankbenutzern den Zugriff zum Betriebsystem zu ermöglichen, ohne daß der Datenbankadministrator besondere Systemrechte (z.B. bei Unix Root-Rechte) haben muß. Er kann aber genauso wenig Benutzer zu einer anderen Applikation zulassen, die auf dem gleichen System installiert ist.

Für den Verantwortlichen eines Bereiches ist es immer wichtig, daß er die Administratoren überwachen kann. Dies ist unter AccessMaster einmal durch die Audits möglich, oder aber auch durch SNMP-Traps an eine Management-Plattform, wie das OpenMaster Framework.

Ein zusätzlicher Sicherheitsgewinn, ist die verschlüsselte Übertragung des Primary Paßwortes, falls auf allen Client-Systemen der Security Data Manager installiert ist. Dies kommt in der Fachhochschule allerdings nicht zum tragen, da die meisten Clients über Standard-Protokolle mit Gateways und Proxies kommunizieren, bei denen eine solche Verschlüsselung nicht vorgesehen ist.

Schlußbetrachtung

Ich betrachte diese Arbeit als erfolgreichen Abschluß meiner drei monatigen Tätigkeit. Mir ist bewußt, daß noch nicht alle Fragen vollends geklärt sind. So ist die Unterstützung von SGI/IRIX 6.x, die ich vorausgesetzt habe noch nicht vorhanden. Sollte eine Unterstützung ausbleiben, verhindert diese nur die Einführung von Single Sign-On. Die Administration der Benutzerrechte stellt kein Problem dar, da dies durch den NIS-Master, ein AIX-System, erfolgt.

Die von mir entwickelten Programme sind noch nicht vollständig. Die Hauptfunktionen sind vorhanden, aber es fehlt z.B. dem FTP-Proxy noch die Unterstützung des Passiven Datentransfers und die Behandlung von Out-of-Band Daten (z.B. CTRL-C während einer Datenübertragung). Für den Telnet-Proxy für Windows-Umgebungen ist eine sinnvolle Erweiterung die Liste der erreichbaren Systeme dynamisch zu erstellen.

Da die Zeit, wie alle Ressourcen, begrenzt ist, habe ich mich auf das Wesentliche beschränkt, um eine abgerundete Arbeit abzugeben.

Die Arbeit hat mich in meiner Überzeugung gestärkt, daß ein Produkt wie AccessMaster, die Benutzeradministration im Labor für Allgemeine Datenverarbeitung erheblich vereinfachen und beschleunigen kann. In Zukunft wäre es durch die zur Verfügung stehenden APIs sogar denkbar, daß die alljährliche Rückmeldung automatisch durch ein Programm erfolgt, ohne das die für die Rückmeldung vorgesehenen Listen manuell erstellt und anschließend manuell mit den Systemen abgestimmt werden müssen.

Anhang

Quelltexte

Die Quelltexte befinden sich im externen Anhang „Quelltexte zur Diplomarbeit“.

Glossar

GINA	Graphical Identification aNd Authentication API : Diese Schnittstelle ermöglicht es Softwareherstellern die Login-Maske von Windows NT durch eine Eigenentwicklung zu ersetzen.
ISM	ISM steht für „Integrated System Management“ und ist die bisherige Bezeichnung des Produktportfolios OpenMaster. Aus Gründen des Markenrechts ist der Name in OpenMaster geändert worden.
NAS	Network Access Server : Router oder Server, der es Benutzern ermöglicht sich über das Telefonnetz in ein Netzwerk einzuwählen.
primary Login	Das Login und Paßwort, das der Benutzer kennt und beim Anmelden am System eingibt.
Proxy	Gateway auf Anwendungsebene, das in die Verbindung zwischen Client und Server eingesetzt wird.
SDM	Security Data Manager : Lokaler AccessMaster-Client, der den lokalen SSO-Cache verwaltet und die Kommunikation mit dem Security Server und dem Workgroup Server durchführt.
secondary Login	Das Login und Paßwort, das auf dem Zielsystem bzw. von der Anwendung benutzt wird.

Security Policy	Umfassendes Sicherheitskonzept für eine Organisation.
Security Server	Der Server, auf dem die zentrale Benutzerverwaltung installiert ist und der die Authentisierung der Benutzer durchführt.
SIB	Security Information Base : Dies ist der Name der Datenbank von AccessMaster, in der die Datenstrukturen die zur Benutzerverwaltung benötigt werden gespeichert sind.
SMB-Protokoll	Session Message Block Protokoll : Protokoll um auf Netzwerk-Ressourcen wie Fileshares und Drucker zu zugreifen. Das Protokoll wurde von Microsoft entwickelt und wird meistens zwischen Windows-Systemen eingesetzt.
SRB	Software Release Bulletin : Ein Dokument des ISM Support Teams, in dem wichtige Voraussetzungen für die Installation der Software, sowie bekannte Fehler, beschrieben sind.
SSO	Single Sign-On : Der Benutzer meldet sich nur noch einmal an und ein Daemon stellt die Logins und Paßwörter für die einzelnen Anwendungen automatisch in die Dialoge ein.
TSB	Technical Software Bulletin : TSBs werden vom ISM Support Team herausgegeben und dienen als Hilfestellungen für die Installation und Konfiguration der Software.
Workgroup Server	Im Sinne von AccessMaster gehören mehrere Workstations immer zu einem oder mehreren Workgroup Servern. Dieser Workgroup Server enthält dann die Audit-

Daten und evtl. die Cache-Dateien. Da das Umfeld des ADV-Labors sich nur auf einen Standort bezieht, ist eine Aufteilung in mehrere Workgoups nicht erforderlich. In dieser Diplomarbeit ist ein Workgroup Server ein Server auf dem Benutzer mit AccessMaster verwaltet werden und der in Zukunft evtl. mal zu einem Workgroup Server im Sinne von AccessMaster werden kann.

Literaturverzeichnis

OpenMaster Dokumentation

ISM Support team
Technical Software Bulletin : Installation of AccessMaster
Groupe Bull, 13. März 1998

Software Release Bulletin : ISM/OpenMaster 4.54
Bull / ISM Post Sales & Support, 9. April 1998

Bull/ISM-BU product line
AccessMaster White Paper
Groupe Bull, 14. Januar 1998

InformationWeek Labs „Virtual Enterprise“
http://pubs.cmpnet.com/iw/670/iwlv.htm
InformationWeek, 23. Februar 1998

Sean Gallagher
Bull: The Secret Is Out, Part 3 of a series on enterprise systems-management products
InformationWeek Labs, 9. März 1998

William Stallings
SNMP, SNMPv2 and CMIP : The Practical Guide to Network-Management Standards
Addison-Wesley Publishing Company, 1993

W. Richard Stevens
Unix Network Programming
Prentice Hall, 1990

David Flanagan
Java in A Nutshell, Deutsche Ausgabe für Java 1.0
O'Reilly/International Thomson Verlag, Bonn 1996

HP OpenView
http://www.hp.com/ http://www.openview.hp.com/

Tivoli
http://www.tivoli.com/

CA Unicenter TNG
http://www.cai.com/

Trusted Information Systems
Firewall Toolkit
http://www.tis.com/

J. Postel, J.Reynolds
RFC 959 : File Transfer Protocol (FTP)
Network Working Group, Oktober 1985

C. Rigney, A. Rubens, W. Simpson, S. Willens
RFC 2138 : Remote Authentication Dial In User Service (RADIUS)
Network Working Group, April 1997

C.Rigney
RFC 2139 : RADIUS Accounting
Network Working Group, April 1997

CCITT, The International Telegraph and Telephone Consultative Commitee
Blue Book, Volume VIII - Fascicle VIII.8, Data Communication Networks
Directory, Recommendations X.500-X.521
International Telecommunication Union, Melbourne 14.-25. November 1988