



Besondere Lernleistung

im Fach Informatik

Thema:



Konzipieren, Programmieren und Dokumentieren einer zentralisierten, plattformübergreifenden Benutzer-Authentifikation zur Verwaltung eines Schulintranet und Konzipieren, Programmieren und Dokumentieren einer Administrationslösung für ein Schulnetzwerk innerhalb eines niedersächsischen Gymnasiums

vorgelegt von

Florian Flachmeier

Martin Flaßkamp

Graf-Friedrich-Schule

Gymnasium des Landkreises Diepholz



Schuljahre 2004/05 und 2005/06

Fach: Informatik

Verfasser: Florian Flachmeier und Martin Flaßkamp

Thema der besonderen Lernleistung:

Konzipieren, Programmieren und Dokumentieren einer zentralisierten, plattformübergreifenden Benutzer-Authentifikation zur Verwaltung eines Schulintranet und Konzipieren, Programmieren und Dokumentieren einer Administrationslösung für ein Schulnetzwerk innerhalb eines niedersächsischen Gymnasiums

Koordinator: Manfred Redetzky, StD

Ausgabetermin: 19.08.2004

Abgabetermin: 31.03.2006

Bewertung der besonderen Lernleistung: Punkte
(siehe auch beigefügtes Gutachten)

.....
Unterschrift der Schüler

.....
Martin Rehermann, StD
(betreuende Lehrkraft)

In einer Welt voller Unsicherheit muss man eine Menge Dinge ausprobieren.
Man kann nur hoffen, dass einige davon funktionieren.

(Douglass North, amerik. Wirtschaftshistoriker und Ökonom)

Inhaltsverzeichnis

1	Einleitung	1
2	Vorüberlegungen	2
2.1	Nutzen einer zentralen Authentifizierungslösung	2
2.2	Ist-Zustand der Datenspeicherung	2
2.3	Vorteile einer Benutzer-Authentifikation	4
2.4	Ergebnisse der Vorüberlegungen	5
2.4.1	LDAP	5
2.4.2	Kerberos	6
2.4.3	OpenAFS	6
2.4.4	Anwendungsfälle	7
3	Umsetzung	8
3.1	OpenAFS	8
3.2	Systemanmeldung	9
3.3	Web-Proxy	10
3.4	Apache	11
3.5	Webfrontend	11
3.6	Shellscript Backend	12
4	Fazit	13
5	Quellenverzeichnis	I
6	Versicherung	III
Anhang		IV
	Schaubilder	IV
	Abb. 1: Anmeldung an Kerberos	IV
	Abb. 2: Derzeitige zentrale Benutzerauthentifizierung an der GFS	IV
	Abb. 3: Mögliche Implementierung an der GFS	V
	PHP mcp.sh API	V
	PHP MCP Functions	VII
	mcp Operationen	IX
	Angefügte CD	X

1 Einleitung

Die IT-Infrastruktur der Graf-Friedrich-Schule gewinnt ständig an Komplexität. Mittlerweile sind über 350 Notebooks, darunter ein mobiler „Computer-Raum“ mit 20 Notebooks auf einem Wagen, Notebook-Beamer-Kombinationen, allgemein nutzbare Schüler-Terminals, diverse Lehrer- und Verwaltungs-Terminals und ein konventioneller Computer-Raum mit 28 Plätzen vorhanden.

Dahinter stehen fünf Server, zwölf W-LAN-Basisstationen für das kabellose Netzwerk, sowie mehrere Kilometer Kabel.

Die Vielzahl an Nutzern mit unterschiedlichen Zugriffsrechten und Anforderungsprofilen stellt einen erheblichen Verwaltungsaufwand bei der Bereitstellung von Diensten dar. Zur Vereinfachung dieser Problematik entwickelten wir ein datenbankbasiertes System, das anhand von Benutzerprofilen die Bereitstellung der zur Verfügung stehenden Dienste verwaltet.

Unser Ziel war es, eine anwenderfreundliche Benutzeroberfläche zu schaffen, die auch ohne Informatik-Kenntnisse einen leichten Zugang zu unserem Administrationssystem bietet.

Als Einstieg in dieses umfangreiche Softwareprojekt nahmen wir Kontakt mit den Rechenzentren verschiedener Universitäten auf und hatten die Gelegenheit, uns im Rechenzentrum der Universität Paderborn bei dem Bereichsleiter Andreas Brennecke über die Grundlagen und die Umsetzung des dortigen Systems zu informieren.

Die Idee zu diesem Projekt entstand während der Mitarbeit in der Administratoren-AG unseres Gymnasiums, die ohne externe Dienstleister das Schulnetzwerk betreut.

Der Schutz vor unberechtigten Zugriffen durch die Benutzer-Authentifikation, die jeden Anwender eindeutig identifiziert und seine Zugriffsrechte definiert, soll die Arbeit der Administration verringern.

2 Vorüberlegungen

2.1 Nutzen einer zentralen Authentifizierungslösung

Anhand der Aufgabenstellung überlegten wir uns zunächst, welchen konkreten Nutzen eine zentrale Authentifizierungslösung bietet und wie man diese innerhalb der GFS umsetzen kann. Da eine Vielzahl von Anwendungsgebieten unter Berücksichtigung der Benutzeridentität existiert, entschieden wir, einen besonders problematischen Schwerpunkt herauszugreifen und durch eine zentrale Authentifizierung zu verbessern. Dieser Schwerpunkt besteht in der zentralen und sicheren Speicherung von Daten durch die Benutzer im Rahmen des Unterrichts oder persönlicher Arbeiten.

2.2 Ist-Zustand der Datenspeicherung

Zurzeit wird die Datenspeicherung durch die Bereitstellung eines CIFS¹-Netzlaufwerks über verschiedene Server, im Falle der Nutzung für Unterrichtszwecke über den Terminalserver „europa“, gelöst.

Dieses Verfahren, das in der Praxis zunächst simpel und einfach erscheint, stellt aber aus mehreren Gründen eine besonders unvorteilhafte Lösung dar:

- Die Daten werden ohne Sicherstellung der Identität des Benutzers gespeichert; somit hat jeder Zugriff auf alle Daten in der Freigabe. Daraus resultiert, dass zum Beispiel ein Einzelner die Möglichkeit hat, die Arbeitsergebnisse einer ganzen Klasse oder eines Kurses innerhalb kürzester Zeit zunichte zu machen.
- Die Daten liegen weiterhin häufig ohne nochmalige Verwendung auf dem Server und – da keine Zuordnung zu Benutzern stattfindet – auch ohne entsprechende Quotierung des Speicherplatzes, wodurch häufig riesige Datenmengen ungenutzt gespeichert werden.
- CIFS ist eine Entwicklung der Microsoft Corporation, die hauptsächlich auf die Verwendung unter Microsoft Windows ausgelegt wurde. Andere

¹ CIFS = „Common Internet File System“; von der Firma Microsoft 1996 eingeführte Erweiterung der Windows-Dateifreigabe SMB („Short Message Block“)

Betriebssysteme können mittlerweile ebenfalls auf dieses Kommunikationsprotokoll zugreifen. Allerdings unterliegt ein Großteil der Protokolle der Entwicklung von Microsoft und ist somit „closed source“. Das bedeutet, dass die Entwicklung vom Hersteller abhängig ist und Veränderungen häufig länger auf sich warten lassen oder gar nicht umgesetzt werden.

- Bei der Verwendung eines Servers unter Microsoft Windows müssen Lizenzen erworben werden, bevor ein Zugriff von Clients auf den Server stattfinden kann. Wird ein Nicht-Server-Betriebssystem von Microsoft eingesetzt, ist hierbei die Zahl der maximalen Clients auf 10 begrenzt.

Die Vorteile dieser Lösung per Freigabe sind im kleinen Rahmen durchaus überzeugend:

- Es ist keine komplizierte Infrastruktur notwendig, es reicht ein einfacher Rechner mit einem Betriebssystem, das eine Freigabe zur Verfügung stellen kann.
- Ein Zugriff durch jeden Benutzer ist unproblematisch möglich, sodass ein klasseninterner Austausch von Arbeitsergebnissen erfolgen kann.

Eine andere praktizierte Lösung, um im Unterricht erstellte Daten speichern zu können, besteht darin, die erarbeiteten Daten per E-Mail an sich selbst zu schicken. Auch hierbei ergeben sich aber gravierende Nachteile:

- Das Versenden der Daten erfolgt meist in Eile am Ende der Stunde, sodass durchaus Fehler auftreten und Daten verloren gehen können.
- Die Internetleitung der Schule wird durch den entstehenden Traffic belastet, da nicht nur die Daten selbst übertragen werden, sondern auch die Webseiten der E-Mail-Anbieter, über die die Mails versandt werden.
- Die Daten werden durch das Versenden nur dem Schüler selber zugänglich. Eine Einsicht durch den Lehrer für eventuelle Ratschläge oder Sicherung von Zwischenergebnissen ist nicht möglich. Es kann auch kein einfacher Austausch zwischen den Schülern stattfinden.

Aber auch diese Lösung bietet einige Vorteile:

- Sie ist einfach. Jeder, der einen E-Mail Account besitzt, hat die Möglichkeit, seine Daten für sich zu sichern.
- Die Daten sind in den meisten Fällen gut gesichert. Große E-Mail-Provider verfügen über gute Backup-Methoden.

Als weitere Möglichkeit besteht noch die Speicherung auf Massenspeichemedien, wie z.B. CD-Rs und Floppys oder USB-Sticks. Auch hier bestehen eindeutige Nachteile:

- Nicht jeder Rechner lässt jedes Medium zu. Sehr alte Rechner verfügen über keine USB-Ports, sodass kein USB-Stick verwendet werden kann. Kaum ein Rechner in der Schule verfügt über einen CD-Brenner, Diskettenlaufwerke kann man bei den neuesten Rechnern auch nicht mehr voraussetzen. Somit ist man gezwungen, eine ganze Sammlung von Backup-Medien mit sich herumzutragen.
- Wie die Lösung des Versendens an die eigene E-Mail-Adresse bietet auch diese Methode keinen Zugriff auf die Daten seitens der anderen Beteiligten am Kurs oder an der Klasse.
- Außerdem besteht ein großer Nachteil in Bezug auf die Datensicherheit, da das Speicher-Medium verloren gehen kann. Somit besteht die Möglichkeit, dass Unbefugte auf die in der Regel zwar meist nicht vertraulichen Daten zugreifen können oder die Daten bei Bedarf nicht verfügbar sind.

Es lassen sich allerdings auch hier einige Vorteile feststellen:

- Die Speicherung der Daten ist überwiegend recht einfach, da sie wie bei normaler Speicherung auf der Festplatte geschieht.
- Der Aufwand, der betrieben werden muss, ist relativ gering; ein Sicherungsmedium an jedem Rechner kann meist vorausgesetzt werden.

2.3 Vorteile einer Benutzer-Authentifikation

In Bezug auf die unter 2.2 genannten Punkte ergeben sich folgende Vorteile speziell für die Datenspeicherung:

- Jeder Benutzer erhält einen persönlichen Ordner, in dem er die Zugriffsrechte von anderen Benutzern frei definieren kann. Somit ist es möglich, den Mitschülern Lesezugriff und dem Lehrer Schreibzugriff für Korrekturen und Ratschläge zu geben.
- Der persönliche Ordner lässt sich auf eine bestimmte Größe beschränken. Somit ist jeder Benutzer gezwungen, ihn selbst zu verwalten.
- Dateien können direkt im persönlichen Ordner wie auf einer normalen Festplatte geöffnet, bearbeitet und gespeichert werden. Außerdem werden sie durch die Sicherungs-Routinen des Systems vor einem Datenverlust geschützt.

Darüber hinaus ergeben sich einige allgemeine Vorteile:

- Die Sicherheit wird erhöht, da die Zugriffskontrolle sich nach dem Bedarf gestalten lässt.
- Begrenzte Ressourcen wie Bandbreite und Speicherplatz können effizienter genutzt werden.
- Die Administratoren werden durch die Auslagerung von Verwaltungsaufgaben entlastet.
- Die zentrale Authentifikation bietet eine Basis für zukünftige Weiterentwicklungen und Erweiterungen von authentifizierten Diensten.

2.4 Ergebnisse der Vorüberlegungen

Nach Internetrecherchen entschieden wir uns dazu, die Datenbank, auf der das System basiert, durch die beiden Dienste *LDAP* und *Kerberos* zu realisieren.

2.4.1 LDAP²

Das *Lightweight Directory Access Protocol*, kurz LDAP, ermöglicht den Zugriff auf einen Verzeichnisdienst. In unserem Fall beinhaltet dieses Verzeichnis die Daten über die Benutzer, zum Beispiel Adresse und Pfad zum Verzeichnis des Benutzers.

² Fredriksson, Turbo: LDAPv3. Internet Dokument: <http://www.bayour.com/LDAPv3-HOWTO.html> [29.03.2006]

Grundsätzlich lässt sich LDAP mit einem Telefonbuch vergleichen, denn es ist darauf optimiert, dass viele Zugriffe erfolgen, aber nur selten etwas geändert wird. Die Daten werden nicht in Zeilen wie in einer Tabelle gespeichert, sondern in *Entries* (\approx Kategorien), die mit Attributen versehen sind. Zum Beispiel könnte der *Entry* „Benutzername“ das Attribut „Max Mustermann“ haben. Die Struktur im Verzeichnis wird als Schema bezeichnet und kann den Erfordernissen entsprechend angepasst werden.

2.4.2 Kerberos³

Die Anwendung *Kerberos* ermöglicht die Authentifikation eines Benutzers in einem Netzwerk gegenüber einem Dienst oder einer Anwendung auf einem Server. Sowohl der Anwender wird gegenüber dem den Dienst bereitstellenden Server authentifiziert, als auch der Server gegenüber dem Benutzer. Für die Anmeldung am *Kerberos*-Dienst fordert der Benutzer, nach Eingabe eines Passworts oder automatisch, ein Ticket, das so genannte *Ticket Granting Ticket* (TGT), an.⁴ Um sich bei einem von Kerberos unterstützten Dienst anzumelden, fordert der Benutzer ein weiteres Ticket vom Kerberos-Server an, mit dem er sich bei dem gewünschten Dienst authentifizieren kann.

Jeder Kerberos-Server ist zuständig für einen bestimmten Bereich, auch *Realm* genannt. Allen zu diesem *Realm* gehörenden Benutzern, Rechnern und Diensten sind eindeutige Namen, die *Kerberos Principals*, zugeordnet.

2.4.3 OpenAFS

Durch einen Besuch im Rechenzentrum der Universität Paderborn, den uns der für LDAP zuständige Bereichsleiter Andreas Brennecke ermöglichte, wurden wir auf *OpenAFS* aufmerksam. Hierbei handelt es sich um eine ursprünglich von IBM entwickelte Software, die ein Netzwerkdateisystem zur Verfügung stellt, das allen eben angesprochenen Bedingungen in Kapitel 2.2 entspricht. Nach eingehender Beschäftigung mit den Details dieses Systems erfuhren wir, dass *OpenAFS* zur Authentifizierung seiner Benutzer auf einen

³ Tung, Brian: Kerberos: A Network Authentication System. Reading: Addison Wesley 1999

⁴ siehe Abbildung 1

Kerberos-Server aufsetzt, welcher im Standardumfang durch eine eigene Implementierung eines *Kerberos IV*-Servers zur Verfügung gestellt wird.

Da *Kerberos IV* gegenüber *Kerberos V* zahlreiche Nachteile, vor allem in Sicherheitsfragen, aufweist, entschieden wir uns, anstatt des *Kerberos IV*-Servers einen *Kerberos V*-Server zu verwenden. Hierbei wählten wir die Implementierung des MIT⁵, das neben der norwegischen Heimdal *Kerberos* Software die größte Verbreitung aufweist. Dies bietet eine große Unterstützung seitens der Anwendungssoftware.

Für unsere Aufgabenstellung ist dieser Aspekt besonders wichtig, da der *Kerberos*-Server das zukünftige Kernstück der Benutzerauthentifizierung darstellt und auch andere Applikationen neben OpenAFS ihre Identifikation hierüber abwickeln sollen.

2.4.4 Anwendungsfälle

Als weitere Vorarbeit legten wir fest, welche zusätzlichen Anwendungsgebiete innerhalb des Einsatz-Bereiches in der Graf-Friedrich-Schule in Betracht kommen, da eine Fülle von Software *Kerberos* unterstützt, aber nicht jede Software sinnvoll in dieser Umgebung angewendet werden kann.

Unsere Überlegungen konzentrierten sich auf die folgenden Anwendungsfälle:

- **Anmeldung am System:** Dies ist der klassische Fall einer zentralen Authentifizierung. Ein Benutzer meldet sich am Rechner mit einer persönlichen Kennung und seinem Passwort an. Die Benutzerdaten sind allerdings nicht lokal auf jedem Rechner gespeichert, sondern im Netzwerk auf einem zentralen Server, auf dem die Daten bei Bedarf sehr einfach geändert werden können und im Netz verfügbar sind, ohne jeden Rechner ändern zu müssen.
- **Persönliche Webseite:** Jeder Benutzer erhält eine an seinen Benutzernamen gebundene Webseite, die er in Eigenarbeit mit Inhalten füllen kann. Somit besitzt er die Möglichkeit, Außenstehenden, die über keinen

⁵ MIT = Massachusetts Institute of Technology

Zugriff auf das AFS Directory, den persönlichen Benutzerordner, verfügen, ebenfalls seine Arbeitsergebnisse oder allgemeine Informationen zu präsentieren.

- Authentifizierter Proxy⁶-Server: Hierdurch wird sichergestellt, dass nur angemeldete Benutzer auf das WWW zugreifen können. Dies ist zwar kein allgemeiner Zugriffsschutz auf das Internet, da nicht alle Dienste über den Proxy-Server geleitet werden, bietet aber die Möglichkeit, durch Identifizierung in den Log-Dateien Missbrauch nachweisen und diesen dann entsprechend ahnden zu können.

3 Umsetzung

Nachdem wir eine konkrete Vorstellung über das geplante Projekt gewonnen hatten, begannen wir mit der Implementierung auf einem laufenden System. Zu diesem Zweck installierten wir einen Testserver, der die Server-Software enthielt, sowie einen Testclient, der als Demonstrationssystem für die konkrete Anwendung durch den Benutzer dienen sollte.

In der Umsetzung entschieden wir uns für *Debian*⁷ *GNU/Linux Sarge* (Version 2.1), das zu diesem Zeitpunkt die aktuelle Stable-Distribution darstellte, also ein stabiles System bot.

3.1 OpenAFS⁸

Die Installation⁹ eines laufenden *OpenAFS*-Servers⁹ gelang erst nach einigen Versuchen, da dieser aus mehreren Server-Programmen besteht, die durch den BOS (Basic Overseer) koordiniert werden und eine sehr komplexe Konfiguration erfordern. Ebenso hatten wir zu diesem Zeitpunkt noch keine Lösung gefunden, *OpenAFS* mit einem *Kerberos V*-Server zu betreiben, so dass wir zunächst den enthaltenen *Kerberos IV*-Server verwendeten. Daraus

⁶ Proxy: Dienstprogramm zur Organisation des Datenverkehrs in Netzwerken

⁷ Debian: GNU/Linux-Distribution, die ausschließlich aus freier Software besteht

⁸ Campbell, Richard; Campbell, Andrew: *Managing Afs: The Andrew File System*. Indianapolis: Prentice Hall 1998

⁹ OpenAFS. Internet Dokument: <http://www.openafs.org/pages/doc/AdminGuide/> [29.03.2006]; Community resource for users of the AFS distributed file system. Internet Dokument: <http://www.central.org/> [29.03.2006]

ergab sich jedoch, dass einige von uns gewünschte Funktionen nicht umsetzbar waren.

Dieses Problem schien jedoch bald behoben, da durch Ken Hornstein vom NCSA¹⁰ eine Software namens *afs-krb5 migration kit* entwickelt wurde, die jedoch beim Kompilieren Fehler ausgab und somit nicht zu verwenden war. Schließlich stießen wir auf ein *Debian*-Paket, das eine Binärdatei dieser Software enthielt und somit unser Problem löste. Es ermöglichte uns zum ersten Mal, auf den AFS-Server per *Kerberos V*-Authentifizierung zuzugreifen und dort Daten abzuspeichern. Hierbei stellten wir fest, welche vielfältigen Möglichkeiten *OpenAFS* bietet, um den Zugriff auf die Dateien zu steuern. Hierfür existiert ein sehr differenziertes System von *Access Control Lists*, mittels derer man für jeden Benutzer und jede Gruppe genaue Zugriffsrechte setzen kann.

3.2 Systemanmeldung¹¹

Nachdem dieser komplexe Teilaspekt weitestgehend abgeschlossen war, begannen wir damit, die Benutzeranmeldung am System unter Linux¹² einzurichten. Diese stützt sich als Authentifizierungsmerkmal auf die *Kerberos-User*¹³ und bezieht die Benutzerinformationen, wie die Position des Home-Verzeichnisses, aus der *LDAP*-Datenbank. Hierfür waren einige Änderungen an der Standardkonfiguration der *Pluggable Authentication Modules*¹⁴ (*pam*) notwendig. Unsere Lösung beinhaltet, dass sich ein Benutzer am System und durch die hierbei übermittelten Daten parallel am *OpenAFS* authentifiziert. Dies kann als echtes Single Sign-On (SSO) angesehen werden, da durch diese Einmalanmeldung ein Benutzer nach einer einmaligen Authentifizierung auf alle Rechner und Dienste, für die er berechtigt ist, zugreifen kann, ohne sich jedes Mal neu anmelden zu müssen.

¹⁰ NCSA: National Center for Supercomputing Applications/University of Illinois

¹¹ Mates, Jeremy: Authenticate Windows to Unix Kerberos. Internet Dokument: <http://sial.org/howto/kerberos/windows/> [29.03.2006]

¹² Garman, Jason: Kerberos: The Definitive Guide. Köln: O'Reilly 2003, S. 141 – 144.

¹³ Heimers, Stefan: OpenAFS, OpenLDAP und Kerberos 5 als Server und Client unter Debian/Sarge. Internet Dokument: <http://www.seismo.ethz.ch/linux/afs/> [29.03.2006]

¹⁴ *pam*: Programmierschnittstelle für Authentisierungsdienste

Ergänzend dazu sollte ein *Windows XP*-Client sich gegenüber *Kerberos* authentifizieren¹⁵, was laut Microsoft möglich sein soll. Die Durchführung wurde dadurch verhindert, dass Microsoft bei der Entwicklung von *Windows XP* ein verändertes *Kerberos*-Protokoll zugrunde gelegt hat und dies mit der *Kerberos V*-Implementation des MIT nicht kompatibel ist. Ohnehin ist die Verwendung der *Kerberos*-Authentifizierung unter Windows problematisch, da hier ein Account existieren muss, auf den der *Kerberos*-Account gemappt wird; im Gegensatz zu Linux kann dies nicht durch die Verwendung von *LDAP* kompensiert werden. Somit müsste auf jedem *Windows*-Client entweder ein einziger Account existieren, auf den alle *Kerberos*-Accounts gemappt würden, was äußerst unsicher ist. Oder jeder *Kerberos*-Account müsste einen entsprechenden lokalen Account unter *Windows* besitzen, was ein nicht unerheblicher Arbeitsaufwand wäre. Der Vorteil der zentralen Speicherung und Administration durch *Kerberos* würde dadurch wieder zunichte gemacht. Somit beschränkten wir uns im weiteren Vorgehen auf die Verwendung des *Kerberos V*-Clients des MIT und des *OpenAFS*-Client in der jeweiligen Win32 Version, wodurch eine erfolgreiche Nutzung des *OpenAFS*-Netzlaufwerks auch unter Windows ermöglicht wird.

3.3 Web-Proxy

Der nächste Schritt war die Implementierung eines authentifizierten Web-Proxies. Hierzu beschäftigten wir uns mit den Möglichkeiten des weit verbreiteten Opensource Web-Cache-Proxys *Squid*. Eine native Anbindung an *Kerberos* seitens *Squid* war leider nicht vorgesehen, so dass wir nach einer Alternative suchten. Nachdem wir herausfanden, dass eine *Kerberos*-Anbindung erst mit Version 3.0 geplant ist, entdeckten wir die Möglichkeit, den Proxy-Server gegen die System-Anmeldung zu authentifizieren und somit über einen Umweg die *Kerberos*-Authentifizierung in *Squid* zu verwenden, da die System-Anmeldung bereits in unser Projekt implementiert war.

¹⁵ Garman, Jason: *Kerberos: The Definitive Guide*. Köln: O'Reilly 2003, S. 173 – 186.

3.4 Apache

Die persönliche Webseite stellte sich als eine äußerst sinnvolle Implementierung heraus, die nach längeren Recherchen durch die entsprechende Konfiguration von *Apache* umgesetzt wurde. Hierfür wurde in der Konfiguration von *Apache* die Position des `public_html`-Verzeichnisses von `/home/*/public_html` auf das entsprechende Verzeichnis im `/afs`-Tree geändert. Dies ermöglichte nun die Veröffentlichung von Informationen über `http` durch jeden einzelnen Benutzer.

3.5 Webfrontend

Zur Administration entwickelten wir eine Benutzeroberfläche in Form eines Webfrontends, die mit jedem gängigen Web-Browser aufgerufen werden kann. Das Frontend ist in der Skriptsprache PHP geschrieben und greift auf ein zentrales Shellsript zu.¹⁶

Es ist uns wichtig, dass sich die Benutzeroberfläche möglichst unkompliziert bedienen lässt und somit keine Einarbeitung der Nutzer erforderlich ist. Außerdem wenden wir das Prinzip der geringstmöglichen Privilegierung (*principle of least privilege*) an, um einerseits aus Gründen der Sicherheit dem jeweiligen Benutzer nur die Rechte zu geben, die ihm zustehen, und andererseits die Benutzung zu vereinfachen. Daher wird in vier verschiedene Benutzergruppen differenziert: Schüler, Lehrer, Manager und Administratoren.

In der Navigation erfolgt eine Aufspaltung der verschiedenen Funktionen. So beinhaltet der Bereich „Menü“ generelle Aufgaben und die das eigene Benutzerprofil betreffenden Punkte. Die Bereiche „Benutzer“ und „Gruppen“ sind für die Benutzer- und Gruppenverwaltung zuständig.

Jedem Benutzer ist es erlaubt, sein eigenes Profil zu ändern und sich die FAQ-Seite (*frequently asked questions*), eine Seite, auf der die Antworten zu häufig gestellten Fragen stehen, anzeigen zu lassen. Die Einträge auf dieser Seite können von Managern und Administratoren über das SQM-Modul (*Support Query Manager*) administriert werden. Dabei ist es möglich, dass Fragen,

¹⁶ siehe 3.6

Probleme und Störungsmeldungen, die über das Hilfe-Formular gestellt wurden, direkt in die FAQ-Seite integriert werden.

Die Gruppe „Lehrer“ hat als besondere Funktion den Klassenbildungsassistenten, der eine Unterstützung dafür bietet, eine größere Anzahl Benutzer einer Gruppe hinzuzufügen. Eine besondere Funktion der „Manager“ ist der Jahresabschluss. Damit ist es möglich, am Ende eines Schuljahres die Bezeichnungen der Gruppen, die für Klassen eingerichtet wurden, entsprechend dem folgenden Schuljahr zu erhöhen. So wird beispielsweise aus der 7a die 8a.

Hervorzuheben ist, dass der tägliche Betrieb ohne regelmäßige Arbeit der Administratoren möglich ist. Die Klassifizierung der Benutzergruppen mit unterschiedlichen Rechten bietet Flexibilität in der Hinsicht, dass die meisten Aufgaben von Benutzern der Gruppen „Lehrer“ und „Manager“ durchgeführt werden können.

Die individuelle Registrierung erfolgt mittels einer Transaktionsnummer, abgekürzt TAN, die an einem dedizierten Rechner im Gebäude der Graf-Friedrich-Schule in Verbindung mit dem Namen generiert wird. Innerhalb von 48 Stunden muss mit dieser TAN die Registrierung von einem beliebigen Rechner aus abgeschlossen werden. Dadurch wird sichergestellt, dass die registrierende Person die Graf-Friedrich-Schule besucht. Außerdem wird im weiteren Verlauf der Registrierung automatisch eine E-Mail an die angegebene Adresse verschickt um zu gewährleisten, dass die E-Mail-Adresse existiert.

3.6 Shellscript Backend¹⁷

Die Verwaltung des mittlerweile sehr komplex gewordenen Systems stellte uns vor eine letzte Herausforderung. In drei unabhängigen Datenbanken, *MIT Kerberos V*, *OpenLDAP* und *OpenAFS* müssen die Benutzerdaten eingetragen und anschließend konsistent gehalten werden, denn alle Datenbanken sind in einer bestimmten Kombination für jeden Einzelnen der eben beschriebenen Dienste notwendig.

¹⁷ Mendel Cooper: Advanced Bash-Scripting Guide. Internet Dokument: <http://www.tldp.org/LDP/abs/html/> [29.03.2006]

So mussten wir eine Lösung finden, die verhindert, dass durch die Änderung einer einzelnen Datenbank das gesamte System inkonsistent wird und möglicherweise fehlerhaft arbeitet. Die Lösung ist eine Eigenentwicklung: ein Shellsript¹⁸, das auf die Administrationswerkzeuge der einzelnen Dienste aufsetzt und an zentraler Stelle eine Schnittstelle zum Verändern der im System gespeicherten Daten vorsieht.

Zunächst wurden hierfür die gewünschten Operationen festgelegt und danach überlegt, welche einzelnen Befehle¹⁹ notwendig sind, um diese Operation durchzuführen.

Aus diesen Überlegungen entstanden die im Anhang aufgelisteten, mittels des Shellscripts möglichen Operationen, die auch ohne das später entwickelte Webfrontend benutzt werden können. Das Shellsript bildet somit eine unabhängige zentrale Einheit, durch die das gesamte System zu kontrollieren ist. Es verwendet für die Operationen, die auf *Kerberos*- oder *AFS*-Programme zugreifen, einen *Kerberos Administrator Principal*, sowie für Zugriffe auf das *LDAP-Directory* den *LDAP Manager Account*. Somit lässt sich auch über die Textkonsole das System administrieren, was äußerst wichtig für direkte Zugriffe auf die Serverkonsole bzw. Remote Administration per Textkonsole ist und eine Schnittstelle für Weiterentwicklungen bietet.

4 Fazit

Im Laufe der Entwicklung unseres Systems hat sich herausgestellt, dass es sich um eine für die Graf-Friedrich-Schule wertvolle Ergänzung der bestehenden IT-Infrastruktur handelt, da hierdurch eine Vielzahl neuer Möglichkeiten entsteht, um das im Netzwerk steckende Potenzial besser zu nutzen. Eine zentrale Authentifizierungslösung bietet den Benutzern mehr Freiheiten und mehr Möglichkeiten die vorhandenen Technologien anzuwenden. Vor allem aber gibt es den Administratoren ein effektives Werkzeug in die Hand, durch das sie das Netz vor unerwünschten Eingriffen schützen können.

¹⁸ Shellsript: eine in eine Datei geschriebene, zusammenhängende Kommandofolge

¹⁹ Heimers, Stefan: OpenAFS, OpenLDAP und Kerberos 5 als Server und Client unter Debian/Sarge. Internet Dokument: <http://www.seismo.ethz.ch/linux/afs/> [29.03.2006]

Unsere Erfahrungen innerhalb der letzten zwei Jahre haben jedoch auch gezeigt, dass eine perfekte Lösung, die alles Machbare einschließt, mit vertretbarem Aufwand nicht zu realisieren ist. Daher haben wir uns bei der Implementierung auf die Aspekte beschränkt, die den größtmöglichen Nutzen für die Benutzer innerhalb der Schule bieten. Das bedeutet, dass unsere Entwicklung durchaus noch Erweiterungen zulässt und im Rahmen eines sich zukünftig wandelnden Anwendungsszenarios flexibel angepasst werden kann.

Allerdings mussten wir nach eingehender Auseinandersetzung mit der Anwendung des Systems erkennen, dass unsere Implementierung lediglich als Proof-Of-Concept²⁰ anzusehen ist. Dies ist zunächst auf die Beschränkung der zu Evaluationszwecken zur Verfügung stehenden Hardware zurückzuführen, die auf ein unter Produktivbedingungen laufendes Fileserver-System, wie der *OpenAFS*-Server, nicht ausgelegt ist. Allein der Fileserver benötigt einen High-End-Server, um die Belastung durch Anfragen, die ein normaler Schulbetrieb verursachen würde, verarbeiten und die Daten sicher speichern zu können. Weiterhin ist auch die Gefahr eines Ausfalls nicht zu vernachlässigen. Ein hohes Maß an Zentralisierung birgt stets die Gefahr einer weit reichenden technischen Störung und stellt somit einen Single-Point-of-Failure²¹ dar. Begünstigt wird dies durch die völlige Zentralisierung unseres Testsystems, da alle Dienste von einem einzelnen Server angeboten werden. Diesem Problem kann man entgegenwirken, indem man zunächst die Dienste auf mehrere Server verteilt und als weiteren Schritt mehrere Server schafft, die parallel den gleichen Dienst anbieten, um eine größtmögliche Redundanz zu erreichen. Dieses Vorgehen wird auf der Abbildung 3 im Anhang beschrieben, das eine konkrete Implementierung für den Produktivbetrieb innerhalb der Graf-Friedrich-Schule zeigt; demgegenüber stellt Abbildung 2 den jetzigen Zustand dar.

Der Vorschlag zur Implementierung sieht vor, dass der Fileserver mittels der *OpenAFS*-internen Mechanismen auf zwei Server verteilt wird, die

²⁰ Proof-of-Concept: englisch "Beweis für die Machbarkeit"

²¹ Single-Point-of-Failure: einzelne Komponente, die bei einem Ausfall den Komplettausfall eines Systems nach sich zieht

beide aus Gründen der Redundanz die Daten enthalten und auf einem RAID5 Array²² sichern.

Neben *OpenAFS* läuft auf diesen Servern ebenfalls verteilt *Kerberos V* sowie *OpenLDAP*; dies stellt eine Sicherung gegen einen zentralen Ausfall dar. Weiterhin ist eine Auslagerung des Proxy-Servers vorgesehen, da, wie bereits dargestellt, *Squid* auf die Systembenutzer aufsetzt. Da es nicht erwünscht ist, dass Benutzer mit Shellzugriff auf den Server zugreifen können, auf dem der Proxy-Server läuft, unterbinden wir dies, sodass Benutzer nicht auf die sensiblen Daten innerhalb der Log-Dateien des Proxy-Servers zugreifen können. Der letzte vorgesehene Server ist der Webserver, der zum einen die *public_html*-Verzeichnisse der Benutzer per *http* freigibt, sowie zum anderen das Webinterface zur Verwaltung ausgibt. Mit dieser Verteilung auf verschiedene Systeme soll die Gefahr eines Totalausfalls verringert werden.

Ebenfalls problematisch ist die notwendige Einarbeitung der Administratoren, die für eine Nutzung des Systems erforderlich ist. Die laufende Verwaltung kann zwar über das Webinterface durchgeführt werden, allerdings ist für besondere Verwaltungsaufgaben oder beim Auftreten von unbekanntem Fehlern eine gute Kenntnis der verwendeten Software und ihrer Eigenheiten notwendig.

Somit lässt sich abschließend sagen, dass eine zentrale Benutzerauthentifizierung innerhalb der Graf-Friedrich-Schule durchaus wünschenswert ist, die Einführung von den Verantwortlichen allerdings sorgsam vorbereitet werden muss, um später schwerwiegende Probleme zu vermeiden.

²² RAID 5: redundantes Speichern von Daten auf einem logischen Laufwerk aus mindestens drei Festplatten

5 Quellenverzeichnis

Literaturquellen:

- Brebeck, Günther; Winkler, Peter: PCs vernetzen. München: Markt&Technik 2000
- Campbell, Richard; Campbell, Andrew: Managing Afs: The Andrew File System. Indianapolis: Prentice Hall 1998
- Carter, Gerald: LDAP: System Administration. Köln: O'Reilly 2003
- Garman, Jason: Kerberos: The Definitive Guide. Köln: O'Reilly 2003
- Keil-Slawik, Reinhard; Brennecke, Andreas; Hohenhaus, Markus: ISIS: Installationshandbuch für lernförderliche Infrastrukturen. Paderborn: Heinz-Nixdorf-Institut 2003
- Olbrich, Alfred: Netze, Protokolle, Spezifikationen. Braunschweig: Vieweg 2003
- Schlüter, Ulrich: Integrationshandbuch Microsoft-Netzwerk. 2. Aufl. Bonn: Galileo Press 2004
- Schüler Duden Informatik. 3. neu bearb. Aufl. Mannheim: Duden Verlag 1997
- Taschenbuch der Informatik / hrsg. von Uwe Schneider und Dieter Werner. 5., neu bearb. Aufl. München:Hanser 2005
- Tung, Brian: Kerberos: A Network Authentication System. Reading: Addison Wesley 1999
- Wong, Clinton: HTTP: kurz & gut. Köln: O'Reilly 2000

Internetquellen:

- Berglund, Johan: AFS Install and Overview. Internet Dokument:
http://www.afsig.se/afsig/space/2004-overview/afs_install_and_overview.pdf [29.03.2006]
- Cooper, Mendel: Advanced Bash-Scripting Guide. Internet Dokument:
<http://www.tldp.org/LDP/abs/html/> [29.03.2006]
- Fredriksson, Turbo: LDAPv3. Internet Dokument:
<http://www.bayour.com/LDAPv3-HOWTO.html> [29.03.2006]
- Haberer, Petra: LDAP verstehen. Internet Dokument:
<http://www.mitlinux.de/ldap/> [29.03.2006]
- Heimers, Stefan: OpenAFS, OpenLDAP und Kerberos 5 als Server und Client unter Debian/Sarge. Internet Dokument:
<http://www.seismo.ethz.ch/linux/afs/> [29.03.2006]
- Heiss, Jason: Replacing NIS with Kerberos and LDAP HOWTO. Internet Dokument: <http://www.ofb.net/~jheiss/krbldap/howto.html> [29.03.2006]
- Mates, Jeremy: Authenticate Windows to Unix Kerberos. Internet Dokument: <http://sial.org/howto/kerberos/windows/> [29.03.2006]
- MIT Kerberos Team: Kerberos: The Network Authentication Protocol. Internet Dokument: <http://web.mit.edu/kerberos/www/krb5-1.4/krb5-1.4.3/doc/krb5-admin/index.html> [29.03.2006]
- OpenAFS. Internet Dokument:
<http://www.openafs.org/pages/doc/AdminGuide/> [29.03.2006]
- Rödel, Jörg: Single-Sign-On mit Kerberos V. Internet Dokument:
<http://www-user.tu-chemnitz.de/~roej/data/kerberos.pdf> [29.03.2006]
- Zuleger, Holger: Kerberos: Single Sign On, Benutzerauthentisierung. Internet Dokument: <http://www.hznet.de/security/kerbeinf.pdf> [29.03.2006]

Sonstige Quellen:

- Münz, Stefan: SELFHTML: <http://de.selfhtml.org>
- Wikipedia - Die freie Enzyklopädie: <http://wikipedia.org/>

6 Versicherung

Wir versichern, dass wir diese Arbeit selbstständig angefertigt haben.

Wir haben keine anderen als die angegebenen Hilfsmittel benutzt. Die Stellen, die wir im Wortlaut oder dem wesentlichen Inhalt nach aus anderen Werken entnommen haben, sind mit genauer Quellenangabe gekennzeichnet. Informationen aus dem Internet haben wir vollständig auf einem Datenträger beigefügt.

Dateien, die wir auf einem elektronischen Datenträger abgeben, haben wir auf Virenfreiheit überprüft.

Diepholz, den 31.03.2006

.....
Florian Flachmeier

.....
Martin Flaßkamp

Anhang

Schaubilder

Abb. 1: Anmeldung an Kerberos

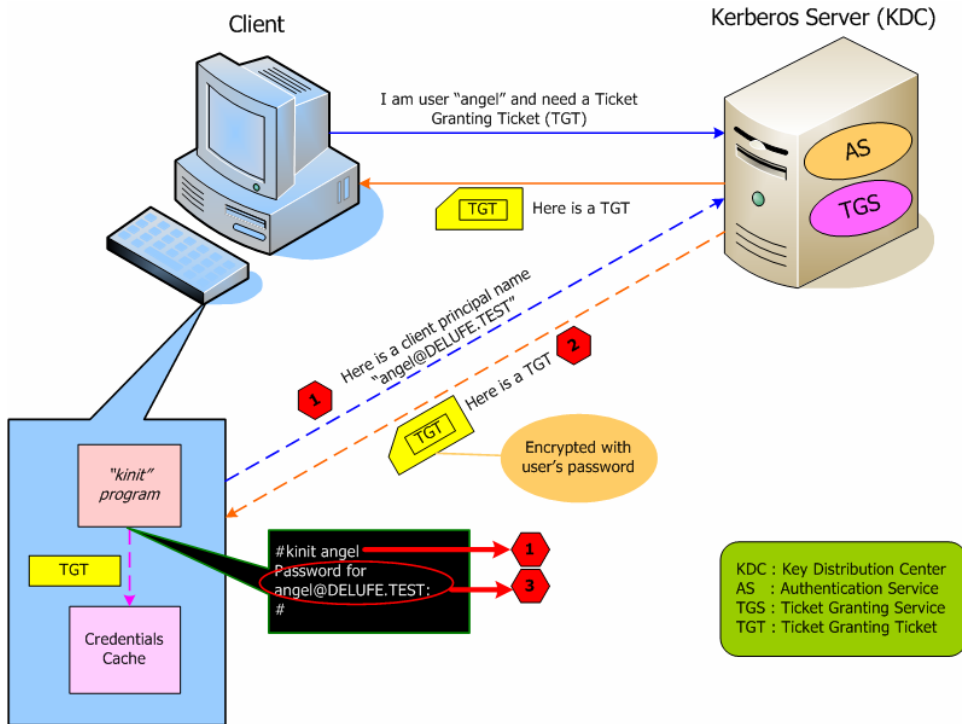


Abb. 2: Derzeitige zentrale Benutzerauthentifizierung an der GFS

Derzeitige Implementierung
einer zentralen Benutzerauthentifizierungslösung
innerhalb des Netzwerkes der Graf-Friedrich-Schule

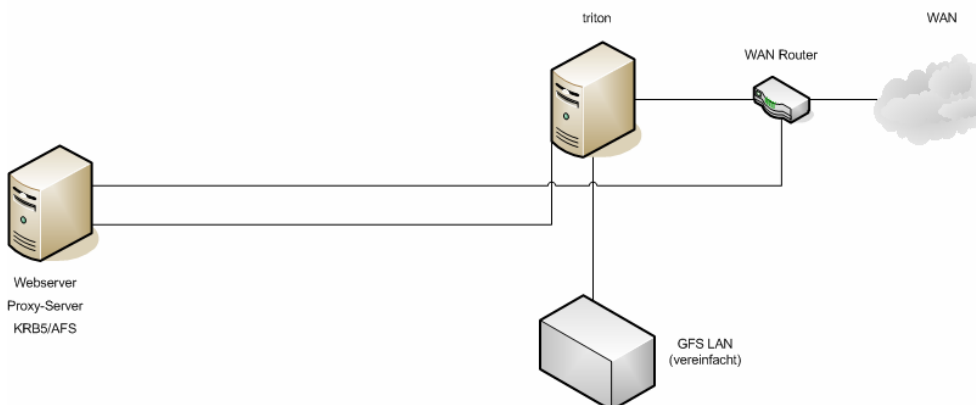
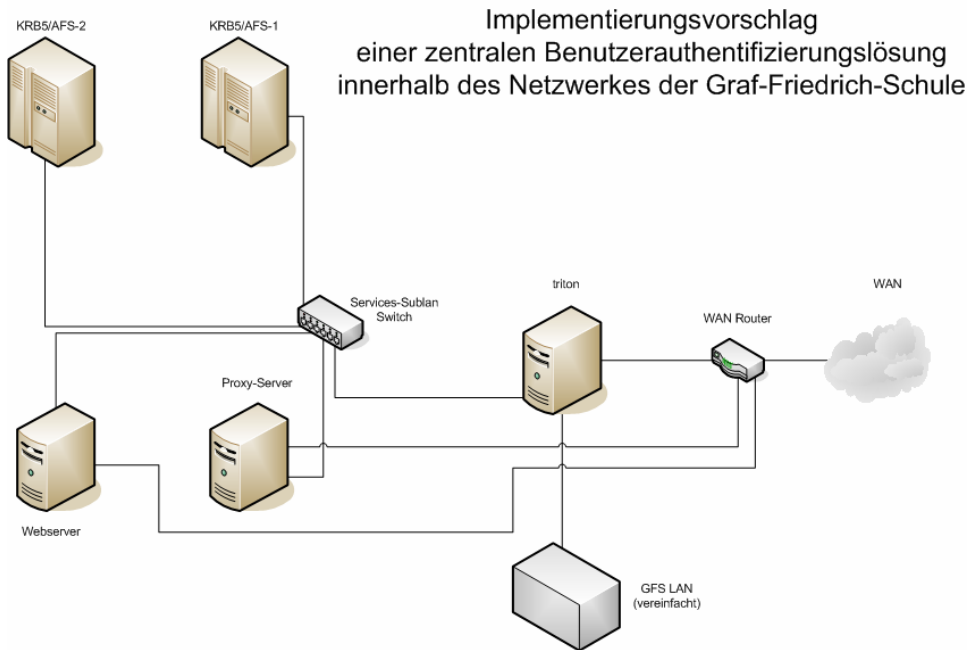


Abb. 3: Mögliche Implementierung an der GFS



PHP mcp.sh API

```
array execcmd(string cmd)
```

Wrapper für cmd Aufruf mit Rückgabe als Array

```
array adduser(string username, string password, string prename,  
string surname, string uid, string gid, string shell,  
string description, string usertype, string fax, string email,  
string phone, string address, string street, string mobile-  
phone)
```

User hinzufügen

```
array deluser(string username)
```

User löschen

```
array moduser(string username, string password, string prename,  
string surname, string uid, string gid, string shell,  
string description, string usertype, string fax, string email,  
string phone, string address, string street, string mobile-  
phone)
```

User modifizieren

```
array listusers(void)
```

Alle User anzeigen im Format <Username>

```
array listuserswithrealm(void)
```

Alle User anzeigen im Format <Username>@<Realm>

```
array userinfo(string username)
```

Informationen über einen User ausgeben

```
array listuserinfo(void)
```

Die Details aller User ausgeben, Standard LDAP Ausgabe, Leerzeilen getrennt

```
array search($crt, $query)
```

Nach verschiedenen Kriterien suchen

```
array chpasswd(string username, string password)
```

Passwort eines Users ändern

```
array addgroup(string groupname, string owner)
```

Gruppe hinzufügen

```
array delgroup(string groupname)
```

Gruppe löschen

```
array listgroups(void)
```

Liste aller Gruppen anzeigen

```
array showgroup(string groupname)
```

Mitglieder einer Gruppe anzeigen

```
array showmembership(string username)
```

Mitgliedschaft in Gruppen anzeigen

```
array showgrpowner(string groupname)
```

Gruppenbesitzer anzeigen

```
array chgrpowner(string owner, string groupname)
```

Gruppenbesitzer ändern

```
array joingroup(string username, string groupname)
```

User in Gruppe hinzufügen

```
array leavegroup(string username, string groupname)
```

User aus Gruppe entfernen

```
bool ismemberof(string username, string groupname)
```

Gruppe auf Usermitgliedschaft prüfen

```
bool isownerof(string username, string groupname)
```

Überprüfen, ob User Gruppenbesitzer ist

```
bool isadmin(string username)
```

Überprüfen, ob User Administrator ist

```
bool ismanager(string username)
```

Überprüfen, ob User Manager ist

```
bool isteacher(string username)
```

Überprüfen, ob User Lehrer ist

```
bool isstudent(string username)
```

Überprüfen, ob User Schüler ist

PHP MCP Functions

```
array deliverlogin(void)
```

Funktion, die den .htaccess Usernamen und das Passwort ausliest und in die Session schreibt.

```
string gen_uname(string prename, string surname, int a, int b)
```

Funktion zur Generierung von möglichen Usernamen. a und b definieren hierbei die Länge des Teils von Vor- und Nachname.

```
array mkuiarray (array array)
```

Funktion, der ein Array mit der LDAP Ausgabe für einen einzelnen Userdatensatz übergeben wird und die dann ein Array zurückliefert, in welchem für jeden Usernamen die einzelnen Werte in Form der script-internen Standardvariablen als Array hinterlegt sind.

```
array getuiarray(string username)
```

Funktion, die die Ausgabe von `userinfo()` in ein Array nach den normalen Variablennamen-Konventionen schreibt.

```
bool is_extant(mixed var)
```

Überprüft, ob eine Variable nicht leer oder gleich 0 ist.

```
bool chkMail(string email)
```

Überprüft eine E-Mail Adresse auf gültiges Format.

```
string topRank(string username)
```

Überprüft, welchen höchsten Userstatus ein Username hat.

```
void authenticate(void)
```

.htaccess "Logout" durch Header rewrite.

```
bool isexisting(string username)
```

Funktion, die einen Usernamen auf Existenz überprüft.

```
bool isgroup(string groupname)
```

Liefert TRUE zurück, falls eine Gruppe existiert, die den übergebenen Namen hat.

```
string retPostGet(string varname)
```

Liefert den Inhalt der POST und GET Array, wenn vorher nicht genau bestimmt werden kann, über welche Methode die Variable übergeben wird.

```
bool isemptygroup(string groupname)
```

Liefert einen booleschen Wert zurück: TRUE, wenn die Gruppe leer ist FALSE, wenn sie Member enthält.

```
array howmanyrequests(string db_server, string db_user, string db_pass, string db_name, string db_table_user_request, string db_table_group_request, string db_supportrequest_table)
```

Liefert die Anzahl von User- und Gruppenanträgen in einem Array zurück.

```
array search_uiarray(string crt, string query)
```

Funktion, die nach einem Query als bestimmtes Kriterium sucht und im Erfolgsfall ein mittels mkuiarray() erzeugtes Array zurückliefert, in dem pro Key ein Array hinterlegt ist, das die einzelnen Userdaten enthält.

```
array parseName(string name)
```

Die Funktion versucht, einen übergebenen Namensstring in Vor- und Nachnamen zu zerlegen. Akzeptiert werden - zumindest liefern diese Formate korrekte Ergebnisse - die Formate <Vorname> <Nachname> sowie <Nachname>, <Vorname>. Bei ersterem besteht allerdings das Problem, dass nicht unbedingt jede Konstellation von Vor- und Nachnamen korrekt geparsed wird, da hier anhand von Leerzeichen die Trennung vorgenommen wird.

```
array getopmail(void)
```

Die Funktion liefert die E-Mail Adressen der Administratoren und Manager zurück.

```
void mailops(string subject, string content)
```

Mail an alle Administratoren und Manager senden (für Statusmeldungen o. ä.).

```
void simplemail(string rcpt, string subject, string content)
```

Mail Funktion mit vorgelegten Headern.

mcp Operationen

adduser	Neuen Benutzer hinzufügen
deluser	Einen existierenden Benutzer löschen
moduser	Einen existierenden Benutzer modifizieren
listusers	Alle Benutzer auflisten
userinfo	Informationen über einen einzelnen Benutzer ausgeben
listuserinfo	Informationen über alle Benutzer ausgeben
search	Nach verschiedenen Kriterien suchen
chpasswd	Passwort eines Benutzers ändern
addgroup	Eine neue Gruppe hinzufügen
delgroup	Eine existierende Gruppe löschen
listgroups	Alle Gruppen auflisten
showgroup	Die Mitglieder einer bestimmten Gruppe anzeigen
showmembership	Alle Gruppen anzeigen, von denen ein bestimmter Benutzer Mitglied ist
showgrpowner	Den Besitzer einer Gruppe anzeigen
chgrpowner	Den Besitzer einer Gruppe ändern
joingroup	Einen Benutzer einer Gruppe hinzufügen
leavegroup	Einen Benutzer aus einer Gruppe entfernen

Detaillierte Hilfen zu jeder einzelnen Operation lassen sich durch folgenden Aufruf anzeigen:

```
/usr/local/bin/mcp help <command>
```

Angefügte CD

Dieser Arbeit wurde eine CD beigefügt, die die selbst erstellten Entwicklungen enthält, sowie alle die Funktionalität von Server und Client betreffenden Konfigurationsdateien. Der Quelltext des Shellscripts, sowie der Quelltext des Webfrontends befinden sich im Verzeichnis „Shellscript“ bzw. „Master Control Program“.

Ebenfalls auf der CD hinterlegt sind die im Quellenverzeichnis aufgeführten Internet-Dokumente in gespeicherter Form.