

Regaining Single Sign-On Taming the Beast

Divyangi Anchan
Ringling School of Art and Design
2700 North Tamiami Trail
Sarasota, Florida 34234
941-309-4150
danchan@ringling.edu

Mahmoud Pegah
Ringling School of Art and Design
2700 North Tamiami Trail
Sarasota, Florida 34234
941-359-7625
mpegah@ringling.edu

ABSTRACT

It has been our effort at Ringling school to provide our campus community with the capability to uniformly access resources across multiple platforms. Empowering the user with a single sign-on capability has multifold benefits. It greatly improves user experience and relieves the user from the burden of remembering multiple user-id and password pairs. On the administrative side, help desk costs are noticeably reduced and security improved, as users are not tempted to 'store' multiple passwords in written form.

In the Fall of 1998 we implemented a single sign-on framework that utilized Sun RPC to synchronize accounts and passwords across multiple systems on the network. Our approach was easy to deploy, did not require any client level software and we did not introduce a single point of failure.

It is our objective to consolidate user administration systems by adhering to a Light Weight Directory Access Protocol (LDAP) based meta-directory model while preserving a common, single end-user authentication information and the coordinated management of user account information. Therefore, we are implementing a new single sign-on system that utilizes LDAP and SUN RPC protocols. Our approach is secure, does not store passwords and does not introduce a single point of failure. Password resetting can be done seamlessly and transparently without the need for additional client software.

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection – *access controls, authentication.*

General Terms: Management, Security, Documentation.

Keywords: LDAP, RPC, Active Directory Service Interfaces (ADSI), Active Directory (AD), Single Sign-On, Account Synchronization, Password Synchronization.

1. INTRODUCTION

Single sign-on is a mechanism whereby a single user-id and password pair will allow a user to access all authorized computer resources in a distributed, multiplatform computing environment, without the need for multiple authentication information.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGUCCS'03, September 21–24, 2003, San Antonio, Texas, USA.

Copyright 2003 ACM 1-58113-665-X/03/0009...\$5.00.

1.1 History

The authors of this paper are aware that single sign-on and its relating issues have grabbed the attention of campus Institutional Technology (IT) units for several years. Several solutions, open source and commercial, are deployed and available in the market. It is not our intention to project our system as a 'first of its kind'; rather, we would like to present our approach towards building the system.

In this section we review major existing single sign-on and password synchronization systems.

1.1.1 Open Source Solutions

The academic technology center at the University of New Hampshire (UNH) [1] presented a paper at the SIGUCCS'02 conference that talks about synchronizing password records between NIS and Active Directory. When compared with our approach, we find that the UNH user account synchronization package is far more complex. Our approach synchronizes password records at the time of account creation. Changes to the password records are propagated on demand as against the UNH approach where synchronization is done every ten minutes. Moreover, the UNH approach uses the entire student database to check for record changes. Our approach requires running our RPC-based client only for the records that need to be updated.

Jon Finke's paper [2] titled 'Embracing and Extending Windows 2000' provides a detailed account of their user account synchronization package. Jon's framework provides an elegant solution but seems to be technically infeasible for our environment due to the use of Oracle as the backend data store. Also, the use of Kerberos for authentication in this architecture makes it unsuitable for us to utilize this approach.

The approach used at Florida Institute of Technology (FIT) [3] is to use the replication mechanism in LDAP to synchronize user accounts records. Although this is an attractive approach, due to issues related to our environment, we decided not to pursue this approach.

1.1.2 Commercial Solutions

Some of the solutions available for single sign-on and password synchronization are: Evidian's AccessMaster [4], Computer Associates's Etrust [5], IBM's Tivoli Global Sign-On [6], Novell's Secure Login [7], Evidian's PortalXpert [8], Entrust's GetAccess [9], Netegrity's SiteMinder [10], RSA Security's ClearTrust [11], Blockade's ManageID Suite [12], Passlogix's v-GO Single Sign-On [13], M-Tech Information Technology Inc.'s ID-Sync [14].

1.2 Multiple Benefits To Single Sign-On

Identity Management has been cited as one of the top ten issues that are of growing importance in modern campus IT environment(s) in the Forth Annual Educause Survey.

“Security and Identity Management is not only on the top ten issues that are of strategic importance, growing in significance, and demanding the campus IT leader’s time, but is now among the top ten in human and fiscal resource consumption, suggesting that campuses are beginning to take action to address this critical challenge.” [15]

The components of identity management include password synchronization, password resetting, single sign-on and access management [16].

There are numerous benefits to single sign-on, including, but not limited to the following:

- The user is concerned with remembering only one set of authentication information. This greatly affects the support provided to users. For example, the number of help desk calls associated with problems such as those related with password resetting requests are greatly reduced.
- Access to several resources with a single user-id and password pair improves user experience.
- The chances of a user committing blunders such as storing passwords in written form out of concern of forgetting them is reduced, which increases security.
- Productivity increases greatly because users do not have to enter authentication information for every resource that they want to access.

2. BACKGROUND INFORMATION

Due to the limitation on the length of this paper, we are unable to provide introductory information on the protocols and technologies referred to in this paper. Interested readers should refer to Directory Services introduction [17], NIS [18], [19], RPC [20], [21], [22], [23], LDAP [17], Active Directory [24], [25] and Active Directory Service Interfaces [26] for more information.

Note: There are several implementations of LDAP and RPC currently available. We use Nebula’s ONC RPC (Windows), SUN RPC (Unix) and OpenLDAP in our framework.

3. CURRENT SINGLE SIGN-ON FRAMEWORK

Our current single sign-on framework is summarized in Figure 1. Windows-based workstations and Mac OS X-based Macintoshes are used in the instructional computing laboratories, which are utilized by faculty and staff members. Besides authentication information required for user logon (Mac OS X uses NIS, Windows uses Active Directory-based domain controllers), the Macintoshes use NIS maps to obtain information for home space mounting via NFS. On Windows, home space mounting is done via Samba shares, and the requisite information is obtained from the user’s profile in Active Directory. Group information is obtained from NIS and Active Directory respectively. File servers, web servers, ftp and telnet servers are all based on Unix. These servers obtain authentication and authorization information from our NIS server. Dialup users are

authenticated via a RADIUS [33] server, which communicates with our NIS server.

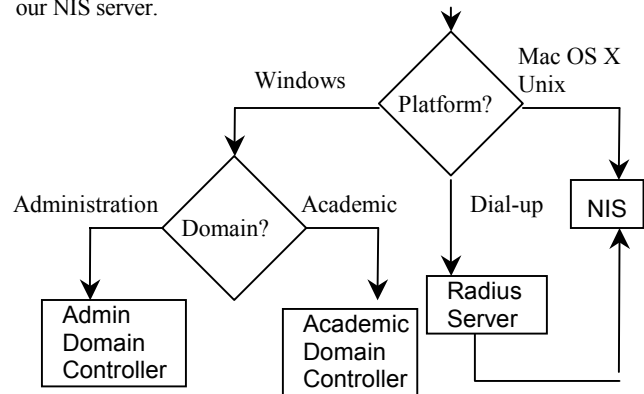


Figure 1 Model directory setup.

3.1 Single Sign-On Package

The single sign-on package consists of the following components:

a. Administrative tasks

This component has client programs that perform tasks such as creating of new accounts, resetting passwords and disabling accounts. These tasks can be performed via a web interface or via a command line.

b. User related tasks

This component has client programs that perform user related tasks such as password changing, mail forwarding, web publishing and print purchasing via a web interface.

c. RPC-based server daemons

This component consists of RPC-based server daemons for file servers (fileserved), NIS server (nisserved), mail server (mailserverd), domain controllers (ntserverd) and LDAP server (ldapserved) that perform the administrative and user related tasks, described earlier, on the server side.

d. Common

This component consists of several common tasks performed on clients and servers (master_create, master_expire, random password generator, UID generator).

3.1.1 Pseudo Code For Administrative Tasks

In this section we summarize the different administrative tasks in simplified pseudo code format. A secret key known to both sides accompanies all communications sent from the RPC client to the RPC-based server daemons. No action is performed on the server side unless the secret key sent by the client is valid.

3.1.1.1 Account Creation

We group the process of account creation into two main functions: create_account and master_create.

```

create_account {
  • Obtain input – First name, Last name, and group id.
  • Make sure the username does not already exist or is not NULL.
  • Generate initial random password for the account.
  • Call function master_create that is the RPC client for creating accounts.
  • Print instruction sheet with requisite information to be handed to user.
}
  
```

```

}
master_create {
    • Ensure that the username received from RPC client does not exist.
    • Create RPC client connections to servers on the NIS server, LDAP server, file servers, mail server, administration domain controller and academic domain controller.
    • Generate a unique user id (UID).
    • Invoke RPC function to add user to the LDAP server.
    • Call RPC functions to populate the passwd, auto.home and mounts.byname maps on the NIS server respectively. Push the NIS maps.
    • Create Samba password entries on the print server (for printer access from the Windows-based workstations) and file servers (for home space access from the workstations)
    • Create links and set quotas on file servers.
    • Set passwords for Samba accounts on print server and file servers.
    • Create home spaces and populate home directories on file servers.
    • Add user on administration and academic domain controllers.
    • If required, make entries for user in corresponding mailing lists.
    • Close all RPC connections.
}

```

3.1.1.2 Account Deletion

We group the process of account deletion into two main functions: delete_account and master_expire.

```

delete_account {
    • Obtain username.
    • Obtain the corresponding password entry from the passwd map.
    • Call master_expire.
}
master_expire {
    • Ensure the user exists.
    • Create client connections to the RPC servers.
    • Remove entry from the LDAP server.
    • Remove user entry from NIS passwd, auto.home and mounts.byname and push NIS maps.
    • Remove Samba password entries from print server and file servers.
    • Remove quota information and links to home directory.
    • Expire home directories.
    • Remake mail tables.
    • Expire mail spool for user.
    • Expire user entry from administration and academic domain controllers.
    • Close RPC connections.
}

```

3.1.1.3 Password Resetting

We group the process of password resetting into two main functions: password_reset and common_changepw.

```

password_reset {
    • Obtain username.
    • Make sure the username exists.
    • Generate a new random password using the random password generator.
    • Call common_changepw to change passwords.
    • Print information sheet for user.
}
common_changepw {
    • Ensure the user exists.
    • Obtain salt values for password encryption.
    • Open RPC connections to RPC servers.
    • Encrypt password.
    • Reset passwords on the NIS server, file servers (Samba), print server (Samba), domain controllers and LDAP server.
    • Close RPC connections.
}

```

3.1.1.4 Modifying Account Attributes

In our current setup, changes to attributes in user records are not expected to occur often. Hence, changes are performed directly on the respective server.

3.1.2 Pseudo Code For User Related Tasks (Changing Passwords)

We summarize below the series of actions performed on the web server when a user changes his/her password via the web interface.

```

action_passwd{
    • Check to see if the username entered exists.
    • Check to see if the 'old password' entered is the same as one obtained from NIS server for the user.
    • Check to see if the two values of new password entered are the same.
    • Check for robustness of password.
    • Call common_changepw to change password as in password_reset above.
}

```

4. MOTIVATION FOR MIGRATION TO LDAP

The first and foremost reason for our migration to LDAP is security. NIS is not only vulnerable to denial of service (DOS) attacks and buffer overflow attacks but is also very inefficient while protecting data. For example, on a Unix machine, guessing the NIS domain and running 'ypcat' to gain access to password records is not impossible. Also, the fact that the NIS daemons can run on unprivileged ports allows an attacker to run replaced version of these daemons and gain access to resources accessible to the NIS daemon [28]. LDAP allows the use of ACL to control access to information, control users that can update LDAP information, etc.

LDAP provides a central location to store any kind of information. This serves as a central repository that provides the convenience of logically structuring the hierarchy according to your requirements.

LDAP supports APIs for writing of programs for manipulation of data, which provides flexibility during implementation.

LDAP uses a database for data storage, which makes access faster for larger number of entries in comparison to NIS, which is based on a flat-file system. Also, LDAP supports hooks for different databases, which provides flexibility.

Use of the 'net' command in our current framework to create entries in Active Directory has a drawback. The 'net' command cannot be used to enter attributes such as e-mail addresses into a user's account. This prevents the use of Active Directory as the authentication service for applications such as Outlook. Initially, we used our existing LDAP server with applications such as Outlook, which increased the LDAP server load when users performed searches on the server. It thus became crucial to integrate the addition of such attributes into user records in the Active Directory via the single sign-on framework.

5. A MODEL SINGLE SIGN-ON SETUP

5.1 Initial Setup

5.1.1 OpenLDAP On Unix Server

The setup of the LDAP server involves the following steps:

1. Download and install the OpenLDAP package [26].

2. Configure a model directory structure. We summarize our model structure below in Figure 2.

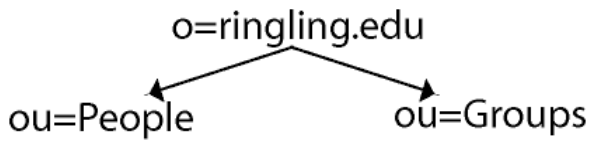


Figure 2 Model directory setup.

3. Extend the OpenLDAP schema to include a new auxiliary class called 'automountInformation'. This new class provides automounting and home space information to Mac OS X clients. This is equivalent to having the mounts.byname map in NIS. The objectclass 'automountInformation' has two attributes types.

macHomeDirectory: This is an XML string specifying the network location of a user's home directory. It has the following format:

```

<home_dir>
<url>
  nfs://fileserver_name/root_to_home_dirs
</url>
<path>userid</path>
</home_dir>
  
```

For example, if the home spaces of all users were in a fileserver called 'fileserver.ringling.edu' under the path '/home/people', then the value of 'macHomeDirectory' for user 'testanchan' would be:

```

<home_dir>
<url>
  nfs://fileserver.ringling.edu/home/people
</url>
<path>testanchan</path>
</home_dir>
  
```

userSharedFolder: This attribute serves two purposes. It specifies to the Mac OS X client the local mount point for home directories, and the location of the user's home space.

For example, if 'userSharedFolder' has a value of '/homespace/testanchan', then the mount point on the Mac OS X client machine for home directories would be '/private/homespace', and testanchan's home directory would be '/private/homespace/testanchan'.

Since the actual mounting happens within the '/private' directory, we create a link from '/homespace' to '/private/homespace' on the client machine.

The OpenLDAP schemas used are available at: <http://www.rsad.edu/~danchan>.

5.1.2 Mac OS X Client Setup

Authentication against a LDAP version 2 server has been explained in detail at [27]. The setup of a LDAP version 3 server is very similar to the setup of a LDAP version 2 server. In addition to the attributes mapped within the 'user' record, also map the attribute 'homeDirectory' to 'macHomeDirectory' and 'NFSHomeDirectory' to 'userSharedFolder'.

5.1.3 Active Directory On Windows 2000 Server

Active Directory is integrated in the Windows 2000 server architecture. The directory structure does not have to be similar to

the one in the LDAP server; the existing default structure can be used.

5.1.4 RPC On Unix Server

Sun RPC is integrated with Solaris 9 hence no additional setup is required.

5.1.5 RPC On Windows 2000 Server

Netbula has implemented Sun RPC (also known as, ONC RPC) This package can be downloaded from [23]. The package contains the required RPC libraries and header files. Also required are 'portmapper.exe' and 'instsvc.exe'. From the command line, do the following to install portmapper [34]:

```
instsrv.exe "Portmap Service" <full path to portmap.exe>
```

5.1.6 Microsoft's Platform SDK and Visual Studio

6.0 Installation

Microsoft's LDAP implementation and ADSI are part of the platform SDK that can be downloaded from [30]. Visual Studio 6.0 needs to be configured such that programs have access to the appropriate libraries and header files. The steps below were followed to prepare the development environment [29]:

1. Go to Tools->Options. Select the Directories tab.
2. Select 'Include' from the drop down box and enter the 'include' directory path for the Microsoft platform SDK. This 'include' directory should be placed before the Visual Studio 6.0 'include' directory. Repeat the same for the 'Library' drop-down option. Make sure that the Microsoft platform SDK 'include' and 'lib' path are placed before the entries for VC 6.0.
3. Go to Project->Settings.
 - a. Modify the compiler 'include' path. Select 'All Configurations' in the 'Settings for' drop down box. Select the 'C/C++' tab and select 'Preprocessor' from the drop down box. Provide the path to the 'include' directory in the platform SDK in the 'Additional include directories' text box.
 - b. Modify the linker options. Select 'Link' tab and select 'input' from the drop-down box. Include wldap32.lib and oncrpc.lib in the list of libraries. Provide paths to the 'lib' directories in the platform SDK installation and in the Netbula ONC RPC installation.

5.1.7 Netscape's LDAP API For C

The OpenLDAP API and Netscape's LDAP API are two of the numerous LDAP API implementations currently available. We have used Netscape's API for development. Download and installation information can be obtained at [17].

5.1.8 Microsoft's LDAP API

The LDAP API used in the RPC-based server daemon is an integral part of the platform SDK. Once the platform SDK has been installed and configured as described earlier, the LDAP API can be used in the Active Directory server programs.

5.2 Program Architecture

Since the use of LDAP APIs is a ‘first time effort’ on our part, we decided to implement the smallest possible functionality to ensure that such an implementation will fit into our existing environment and meet our requirements. The details of the account management package are given below:

5.2.1 RPC Specification Files

The RPC specification files in the Active Directory and OpenLDAP servers define a single procedure to be implemented. Also defined is a C-style structure for passing user information from the client to the server. Minimal attributes are included and constraints such as length of usernames, etc have been ignored for simplicity.

RPC specification file in Active Directory:

```
pdccprog.x
struct PDCUser{
    string givenName<>;
    string sn<>;
    string displayName<>;
    string userPrincipalName<>;
    string cn<>;
    string dn<>;
    string sAMAccountName<>;
    string userPassword<>;
    string profilePath<>;
    string homeDirectory<>;
    string homeDrive<>;
    string scriptPath<>;
};
program PDCRPCPROG {
    version PDCRPCVER {
        int pdc_add_newuser(PDCUser) = 1;
    } = 1;
} = 0x20000001;
```

RPC specification file in OpenLDAP:

```
ldapprog.x
struct LDAPUser{
    string cn<>;
    string sn<>;
    string uid<>;
    string dn<>;
    string uidNumber<>;
    string userPassword<>;
    string gidNumber<>;
    string loginShell<>;
    string homeDirectory<>;
    string NFSHomeDirectory<>;
};
program LDAPRPCPROG{
    version LDAPRPCVER {
        int ldap_add_newuser(LDAPUser) = 1;
    } = 1;
} = 0x20000002;
```

5.2.2 Implementation Of Client and Server Programs

- Run ‘rpcgen’ on the Windows and Unix side to generate the header files (pdccprog.h and ldapprog.h), xdr files (pdccprog_xdr.c and ldapprog_xdr.c), client (pdccprog_clnt.c and ldapprog_clnt.c) and server (pdccprog_svc.c and ldapprog_svc.c) stubs. On the Unix side you will need to run ‘rpcgen’ for both pdcc.x and ldap.x since we run our client from the Unix side and that requires both the client stubs.
- Implement the client and server programs.

Implementation notes:

- a. On the Windows side RPC-based server daemon implementation, you should include the following header files (from the platform SDK and not from Visual Studio 6.0): windows.h, rpc/rpc.h (from the ONCRPC download), winldap.h, and pdccprog.h.
- b. On the Unix side RPC-based server daemon implementation, you should include rpc/rpc.h, ldap.h, and ldapprog.h.
- c. We run our RPC client Solaris platform and the implementation includes rpc/rpc.h, ldapprog.h, and pdccprog.h.
- d. In our implementation of initial account creation, random passwords are generated on the client side and passed on to the LDAP RPC server in encrypted form with the user records. Passwords in user records sent to the Active directory RPC server are in clear text.
- e. On the Windows side, setting/ changing passwords in user records using LDAP protocol requires the use of 128 bit SSL connection [31]. However, as of this implementation, we have decided not to use this method for password setting/ changing. The ‘net user’ command is used in the implementation to update passwords.
- f. The use of the ‘net’ command to update Active Directory in our implementation requires passwords to be sent in clear text over the network. Typically this is a major security risk. In our switched, flat network, that is not a major concern as sniffing network communications is not as simple as it would be in a typical network.

5.2.3 Compilation Of RPC Server Daemons

- Active Directory RPC server: The project package includes pdccprog.h, pdccprog_xdr.c, pdccprog_svc.c, and pdccprog_svc_impl.c. Before compiling this project, make sure that the necessary libraries and include files have been used in the setup as specified in the ‘Initial Setup’ section.
- LDAP RPC server: The project package includes ldapprog.h, ldapprog_xdr.c, ldapprog_svc.c, and ldapprog_svc_impl.c. A sample ‘makefile’ would include:

```
gcc -Wno-implicit -Wno-unused -Wno-return-type
-I <ldap include directory> -L <ldap libraries
directory> -lrpcsvc -lnsl -lsocket -lldap50
ldapprog_svc.c ldapprog_xdr.c ldapprog_svc_impl.c
-o <output file name>
```
- While compiling the Unix side server daemons and client, it is required that the following libraries be passed to the linker: libnsl.so, librpcsvc.so, libsocket.so and libldap50.so.

5.2.4 Compilation Of RPC Client

The project package includes ldapprog_clnt.c, ldapprog_xdr.c, pdccprog_clnt.c, pdccprog_xdr.c, and client_impl.c. The same libraries are required to be passed to the linker here as in the earlier section for servers. A sample ‘makefile’ would include the following:

```
gcc -Wno-implicit -Wno-return-type -Wno-unused -lnsl
-I <ldap include dir> -L <ldap libraries dir> -lsocket -
lrpcsvc -lldap50 ldapprog_clnt.c ldapprog_xdr.c
pdccprog_clnt.c pdccprog_xdr.c client_impl.c -o <output
file>
```

6. FUTURE MIGRATION PLANS

With the successful implementation of the account creation module, we are now at a stage where the rest of our single sign-on framework can be migrated from NIS to LDAP. Besides migration of all functionalities in the existing framework, some of the issues requiring our attention are:

- Migration of information from NIS maps to LDIF formatted files will mostly be performed using scripts obtained from [32].
- The current authentication mechanism between the RPC client program and the server daemons are via the use of a secret key known to both ends. Although all communications occur within our network, we are considering the use of more secure mechanisms to send data over the network.
- Besides authentication and authorization of users, we use NIS to perform system authentication for NFS mounts and automounting. We have to extend our directory structure to include these capabilities.

7. ACKNOWLEDGMENTS

The authors would like to thank all member of the IT team at the Ringling School of Art and Design who provided valuable input in the planning and implementation of our model LDAP-based single sign-on framework. We would also like to thank Kervin Pierre (Network Administrator, FIT) and Curtis Robinson (Systems Analyst, FIT) for their help during our testing of the FIT single sign-on package and Apple Computer for their support.

8. REFERENCES

- [1] David Blezard and Jerry Marceau, *One User, One Password: Integrating Unix Accounts and Active Directory*, <http://at.unh.edu/siguccs/p078-blezard.html>, 2002.
- [2] Jon Finke, *Embracing and Extending Windows 2000*, <http://www.rpi.edu/~finkej/Papers/LISA2002-Embrace.pdf>, 2002.
- [3] Kervin Pierre, *LDAP Account Synchronization Project*, <http://acctsync.sourceforge.net/>, 2002.
- [4] Evidian, *AccessMaster*, <http://www.evidian.com/accessmaster/about/>.
- [5] Computer Associates, *ETrust Identity Management*, <http://www3.ca.com/Solutions/SubSolution.asp?ID=4347>, 2003.
- [6] IBM, *Tivoli Global Sign-On*, <http://www.ibm.com/software/tivoli/products/gso>, 2003.
- [7] Novell, *NSure*, <http://www.novell.com/solutions/nsure>, 2003.
- [8] Evidian, *PortalXpet*, <http://www.evidian.com/portalxpert>.
- [9] Entrust, *GetAccess*, <http://www.entrust.com/getaccess/index.htm>, 2003.
- [10] Netegrity, *SiteMinder: IdentityManager*, <http://www.netegrity.com/products/products.cfm?page=SMoveview>.
- [11] RSA Security, *ClearTrust*, <http://www.rsasecurity.com/products/cleartrust/datasheets/dsclrtrust.html>, 2003.
- [12] Blockade, *ManageID Suite*, <http://www.blockade.com/products/index.html>, 2003.
- [13] Passlogix, *v-GO Single Sign-On*, <http://www.passlogix.com/>, 2003.
- [14] M-Tech Information Technology, Inc., *ID-Synch*, <http://www.idsync.com>, 2003.
- [15] Grant Crawford, Rudy Julia and The Educause Current Issues Committee, *Fourth Annual Educause Survey Identifies Current IT Issues*, <http://www.educause.edu/ir/library/pdf/EQM0322.pdf>, 2003.
- [16] Rutrell Yasin, *What is Identity Management*, http://www.infosecuritymag.com/2002/apr/cover_casestudy.shtml, 2002.
- [17] The OpenLDAP Project, *OpenLDAP 2.1 Administrator's Guide*, <http://www.openldap.org/doc/admin21/intro.html>, 2003.
- [18] Hal Stern, Mike Eisler and Ricardo Labiaga, *Managing NFS and NIS*, 2nd Edition, o'Reilly.
- [19] Thorsten Kukuk, *The Linux NIS (YP)/NYS/NIS+ Howto*, <http://www.linux.org/docs/ldp/howto/NIS-HOWTO/>, 2002.
- [20] John Bloomer, *Power Programming with RPC*, O'Reilly.
- [21] Dave Marshall, *Programming in C: Remote Procedure Calls*, <http://www.cs.cf.ac.uk/Dave/C/node33.html>, 1999.
- [22] Cory Vondrak, *Remote Procedure Call: Software Technology Review*, http://www.sei.cmu.edu/str/descriptions/rpc_body.html, 1997.
- [23] Netbula, *ONC RPC for Windows*, <http://www.netbula.com>, 2003.
- [24] *About Active Directory*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/ad/about_active_directory.asp.
- [25] *Win2ktech, Windows 2000 Active Directory FAQs*, <http://www.win2ktech.com/faq/faqs2k/actdirfaq.htm#activedirectory3>.
- [26] *Active Directory Service Interfaces Overview*, <http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp>.
- [27] *Directory Services*, http://www.macoslabs.com/documentation/directory_services/intro.html.
- [28] Kristy Westphal, *NFS and NIS Security, 2001*, <http://www.securityfocus.com/infocus/1387>.
- [29] Gil Kirkpatrick, *Active Directory Programming, SAMS*.
- [30] Microsoft MSDN, *Platform SDK Download*, <http://msdn.microsoft.com/downloads/sdks/platforms/platform.asp>.
- [31] Microsoft Knowledge Base, *Description of Password-change Protocols in Windows 2000*, <http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/q264/4/80.ASP&NoWebContent=1>, 2002.
- [32] PADL, *MigrationTools*, <http://www.padl.com/OSS/MigrationTools.html>, 2003.
- [33] *Definition of RADIUS*, <http://www.webopedia.com/TERM/R/RADIUS.html>.
- [34] *Definition of Portmapper*, http://www.usenix.org/publications/library/proceedings/sec98/full_papers/paxson/paxson_html/node21.html.