Case Study and Tutorial: HTTPS Reverse Proxy and Authentication with LDAP

TR-3361 | Niall Doherty | January 2005

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Abstract

Presented here is a case study on how the NetCache® product was deployed as a reverse proxy device, introducing an extra layer of security between clients and the Web server.

The topics of HTTPS reverse proxy and authentication with LDAP are discussed in some detail. Configurations are described for the graphical user interface (GUI) and command line interface (CLI) for NetCache 5.6.2.

Table of Contents

1 Deployment Overview

1.1 Business Requirements

A company needs to provide partners with fast and secure access to sensitive documents and specifications regarding joint development projects. The public Internet is to be used as the medium of communication. Authentication is required so that only authorized users can access the information. The communication sessions must be encrypted so that unauthorized users cannot eavesdrop on the information in transit.

1.2 Technical Requirements

Clients must securely provide credentials to access content on a Web site. The credentials are validated with a user account database and used to determine what information may be accessed.

The network within the data center is considered a trusted environment. A reverse proxy device is required within the DMZ to provide an additional layer of security for the data center. This reverse proxy device terminates client connections directly. New connections are initiated by the reverse proxy device to the Web server on the internal network.

The requirements are summarized below:

- Require clients to access Web server through trusted intermediary device
  - Introduce reverse proxy device (application-level proxy) in DMZ
- Use HTTP over SSL (HTTPS) as the client access protocol
- Use HTTP as the server access protocol
- Require credentials from certain users to access content
  - Validate credentials with Windows® 2000 user account database
  - Only short-form username should be required (no domain name)
- Pass username to application on origin Web server

1.3 Authentication Design

NetApp Technical Report 3336 [TR-3336] describes how NetCache can authenticate users over HTTP [RFC-2616]. Two main methods are used for authentication when Active Directory is used for the user account database:

- Request credentials from end users via HTTP Basic [RFC-2617], then validate them using LDAP

- Request credentials from end users via NTLM over HTTP, then validate them using the NetLogon method

HTTP Basic (and LDAP) was chosen instead of NTLM over HTTP (and NetLogon):

1. HTTP Basic causes the password to be sent in plain text. However, this is not a concern if the entire communication session is protected using SSL, as is the case in this deployment.
2. NTLM provides the possibility of "automatic logon" where users do not have to enter credentials: the OS will supply them automatically. However, if the user is not in the same domain (network) as the server, automatic logon is not performed. In this deployment, users would be prompted with either method.
3. For NetLogon validation, the current NetCache versions require users to provide a domain name in conjunction with their username, and this is undesirable.

Figure 1 below shows the architecture of the solution.

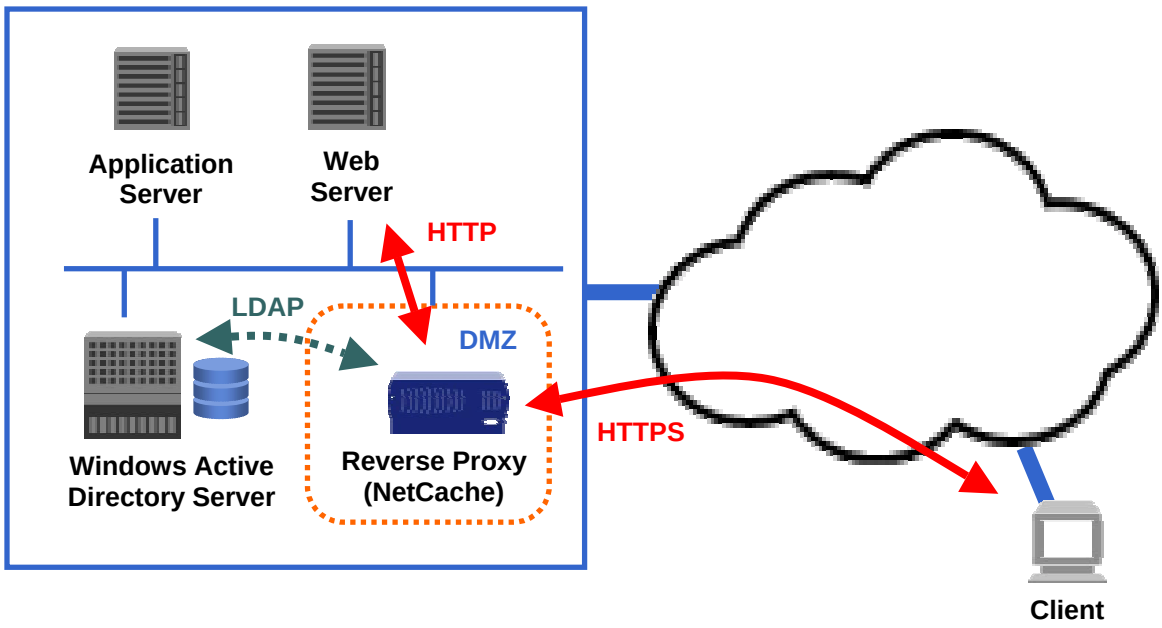

Figure 1) Solution Architecture.

2 SSL and HTTPS Configuration

2.1 Defining SSL Parameters

An *SSL context* is a set of parameters that are applied to an SSL session between a client and a server.

The appropriate options are found in the GUI in this section:

- Setup | Authentication | SSL Contexts

NetCache requires the following parameters to be specified for an SSL context:

- Name
  - This is used to refer to the context in access control lists (ACLs)
- User Certificate
- Key File
- Direction
  - This states whether the context is for SSL connections made to the NetCache, or for SSL connections initiated by the NetCache
- Authentication
  - This determines whether a certificate sent by an SSL peer should be examined for authenticity
- Protocols
- Strength
- Ciphers
- Session Cache
- Session Timeout
- Root Certificates

In this deployment, a context is defined for connections initiated by clients (inbound). Defining multiple inbound SSL contexts would allow different certificates to be associated with different connections (and therefore applications).

SSL authentication is disabled, since validation of client certificates is not required: authentication is performed by requesting a username and password from the client.

An example CLI configuration is:

```
config.ssl.contexts = \\
app1_inbound on certfile.pem keyfile.pem inbound off
ssl2,ssl3 all default 5 300 default
\\
```

A certificate—and corresponding key file—can be generated on NetCache itself or uploaded from another source. The certificate contains a public key that is used by clients to encrypt a symmetric session key. The key file contains the private key that is used by NetCache to decrypt the encrypted data sent by the client.

NetCache protects the key file, and it is unavailable for administrators to download. For this reason, if a certificate is to be signed by a trusted authority, it is recommended that the certificate and key file not be generated by NetCache. This allows administrators to back up the certificates and key files before uploading them to NetCache.

2.2 Accepting Incoming HTTPS connections

The appropriate options are found in the GUI in this section:

- Setup | HTTPS | General

The options are named:

- HTTPS Enable
- HTTPS Proxy Ports
- HTTPS Inbound Context ACL Enable
- HTTPS Inbound Context ACL

An example CLI configuration is:

```
config.https.enable = on
config.https.ports = 443

config.https.inbound.enable = on
config.https.inbound.contexts = \\
context “app1_inbound” any
\\
```

It may be desirable to accept connections on multiple ports using different contexts and providing access to different applications. An example CLI configuration is:

```
config.https.enable = on
config.https.ports = 443 444

config.https.inbound.enable = on
config.https.inbound.contexts = \\
context “app1_inbound” cache-port 443
context “app2_inbound” cache-port 444
\\
```

2.3 HTTPS-to-HTTP Acceleration

The appropriate options are found in the GUI in these sections:

- Setup | HTTPS | Web Server Acceleration | General <TAB>
- Setup | HTTPS | Web Server Acceleration | Acceleration Rules <TAB>

The options are named:

- Web Server HTTPS Accelerator Enable
- Enable HTTPS Monitor
- Web Server HTTPS Acceleration Rules

Below is an example CLI configuration where multiple applications are available:

```
config.https.acceleration.enable = on
config.https.acceleration.monitor.enable = on

config.https.acceleration.rules = \\
netcache-ip1 443 * * WebServer-ip1 80 * *
netcache-ip1 444 * * WebServer-ip2 80 * *
netcache-ip2 443 * * WebServer-ip3 80 * *
\\
```

NetApp Technical Report 3071 [TR-3071] discusses Web site acceleration in detail. The latter portions of section 3.3 describe how the acceleration rules are constructed for DNS-based Web site acceleration (the deployment used here).

NetApp Technical Report 3309 [TR-3309] discusses network configurations for common NetCache deployments. Section 2 contrasts forward and reverse proxy deployments.

3 Requesting Credentials

3.1 Types of HTTP Authentication

A client is challenged for credentials by being sent a specific HTTP header in response to its initial request. There are two types of HTTP authentication, which are very similar in terms of protocol interactions:

- **Proxy authentication.** Clients present credentials to a proxy in order to gain access to resources on Web sites beyond that proxy. The response appears as follows:

```
HTTP/1.1 407 Proxy Authentication Required
Proxy-Authenticate: Basic “realm=xyz”
```

- **WWW authentication.** Clients present credentials to a Web site in order to gain access to a resource on that Web site. The response appears as follows:

```
HTTP/1.1 401 Proxy Authentication Required
WWW-Authenticate: Basic “realm=xyz”
```

If a client is unaware of the presence of a proxy, it will not send credentials if it receives a response indicating that “proxy authentication” is required. Clients are unaware of the presence of a proxy for the following two types of deployments:

1. Transparent redirection
2. Reverse proxy

This deployment is of the latter type. For authentication to succeed in either of these deployments, NetCache must indicate that Web site authentication is required.

The appropriate option is found in the GUI in this section:

- Setup | Authentication | General

The option is named:

- Transparent-mode Authentication

The appropriate CLI configuration is:

```
config.auth.trans_auth = on
```

3.2 Controlling Authentication

It is possible to configure NetCache to require authentication for every request.

The appropriate option is found in the GUI in this section:

- Setup | Authentication | General

The option is named:

- Authenticate Client Requests Using These Protocols

This method is not recommended. Instead, access control lists (ACLs) should be used, since these allow finer control.

The appropriate options are found in the GUI in this section:

- Setup | Access Control | Access Control Lists | Access Control Lists <TAB>

The options are named:

- Enable ACLs
- HTTPS ACL

Below is an example CLI configuration that requires all users to authenticate:

```
config.acl.enable = on

config.https.acl = \\
auth any
\\
```

3.3 Passing a Username to an Application Server

In situations where NetCache needs to contact a Web server to satisfy a client request, it has the ability to include the username that a user authenticated as in its HTTP request. This is done by inserting an HTTP request header. The name of the header is configurable, and its insertion is controlled by ACLs.

The name of the header can be configured in the GUI in this section:

- Setup | HTTP | General

The option is named:

- HTTP User Name Header

An example CLI configuration is:

```
config.http.user_header = X-User-Name

config.https.acl = \\
auth any
user-header any
\\
```

The header will be inserted into every HTTP request for successfully authenticated users when the ACL is evaluated to be true. This is a nonstandard header, and a Web server must be configured appropriately to make use of it.

Note that for Microsoft authentication methods (Kerberos or NTLM), authentication is performed only at the beginning of the connection. NetCache stores the username and inserts it for each request made to the Web server, as appropriate.

3.4 Groups and Protocol Access Permissions

NetCache maintains a local user account database. By default, this database contains a user named “admin” who is a member of a group named “NetCache.” Users are allowed to use particular access methods (protocols) based on their membership in local groups. A default set of permissions is applied if a user is not a member of a local group. By default, HTTPS access is not allowed.

The appropriate option is found in the GUI in this section:

- Setup | Authentication | Groups | General <TAB>

The option is named:

- Default Access Permissions

The following example CLI configuration shows the default settings:

```
config.auth.groups_default_perms =
ftp,gopher,http,tunnel,nntp,rtsp,mms
```

Below is an example CLI configuration allowing only HTTPS client access:

```
config.auth.groups_default_perms = https
```

4 Validating Credentials: Windows 2000 and LDAP

4.1 Defining the Validation Method

NetCache has the ability to validate credentials sent by a client with a number of user account databases. Typically, an administrator account for NetCache is defined in the local database, and this database is searched before alternatives. In this deployment, Microsoft Active Directory is also queried using LDAP.

NetCache has the ability to "remember" which credentials were successfully validated. It can store these for some time (until a TTL expires) in order to reduce the load on the authentication server.

The appropriate options are found in the GUI in this section:

- Setup | Authentication | General

The options are named:

- Authentication Checking Order
- Authentication Server Cache TTL

Below is an example GUI configuration for the first option:

1. Appliance User Database
2. LDAP
3. None
4. None

An example CLI configuration is:

```
config.auth.authorder = \\
local ldap
\\

config.auth.cache_duration = 3600
```

4.2 Accessing the LDAP Directory

4.2.1 Overview

LDAP provides a method of accessing a tree-structured information directory. Entries in the tree (such as user and computer accounts or group objects) consist of attribute name and value pairs. One or more of these attribute values, which must be unique among all entries at that level in the tree, defines an entry's *relative distinguished name* (RDN). Entries are normally referred to by their *distinguished name* (DN), which is a concatenation of the RDNs of the sequence of entries from a particular entry to the root of the tree. An example directory entry—with minimal attributes—is shown below:

```
dn: uid=jblack, OU=division, O=company
uid: jblack
gid: division
mail: jblack@company.com
cn: Joe Black
sn: Black
password: xyz123
```

There are two primary versions of the protocol: version 2 [RFC-1777] and version 3 [RFC-2251] (which includes all elements of version 2). Significantly, the structure of the directory itself is not defined: it is, instead, vendor-dependent.

In order to perform operations on the directory (search, modify, etc.) the LDAP client must first "bind" to the LDAP server. The b*ind operation* initiates the protocol session and allows the client to pass credentials to the server: the client may bind either anonymously (no credentials) or as an entry in the directory. Note that a client may perform any number of bind operations during a protocol session to change identify (may affect authorization).

4.2.2 Microsoft Active Directory

Windows 2000 introduced Active Directory (AD): this database of network information includes user, computer, and group account data. Access is provided primarily through LDAP.

While some LDAP servers may require special privileges to browse the directory, all users in Windows—by default—have the privileges necessary to perform searches on all entries under their branch of the tree. This means that if all users are in the same branch, any regular user may bind and perform searches on any other user.

Active Directory is tightly bound to DNS, and the structure of the tree follows the domain naming of the environment. For LDAPv2, Microsoft states that each component of the domain be referenced using the “DC” attribute name. Organizational units within the structure can be referenced using the “CN” (common-name) attribute. For example:

```
DN = CN=Joe Black, CN=users, DC=division, DC=company, DC=com
```

4.2.3 NetCache Configuration

When configuring NetCache to access an LDAP directory, it is necessary to specify the version to be used as well as the location in the tree where the search should begin (the “Base Distinguished Name”, also referred to as the *baseObject*.) The object to bind as must also be specified.

The appropriate options are found in the GUI in this section:

- Setup | Authentication | LDAP | General <TAB>

The options are named:

- LDAP Enable
- LDAP Servers
- Bind Distinguished Name
- Appliance LDAP Password
- Base Distinguished Name
- LDAP Protocol Version

An example CLI configuration is:

```
config.auth.ldap.enable = on

config.auth.ldap.servers = \\
ldap-server-ip:389
\\

config.auth.ldap.binddn = \\
cn=user1,cn=users,DC=sub-domain,DC=domain,DC=com
\\

config.auth.ldap.bindpass = user1-password

config.auth.ldap.basedn = \\
cn=users,DC=subdomain,DC=domain,DC=com
\\
```

```
config.auth.ldap.version = 2
```

4.3 Searching for Users and Groups

4.3.1 Overview

A s*earch operation* allows a client to read attributes from entries in the directory. In the request, the client states at what point in the tree to begin the search (*baseObject*), what conditions are necessary to satisfy the search (*filter*), and what the response from the server should contain (*attributes*).

There are two common methods of providing group information for users.

1. **Attribute of User Object.** The entry in the directory for the user may contain an attribute listing group membership information.

2. **Group of Unique Names Objects.** The directory may contain entries for individual groups (objectClass=GroupOfUniqueNames). Each entry would contain multiple attributes listing the users that are members of the group.

4.3.2 Microsoft Active Directory

Active Directory stores the logon username of a user account in the attribute named "`samAccountName`" in the account's entry in the directory. The attribute "`memberOf`" can hold group membership information.

GroupOfUniqueNames objects can hold the DN of users as the values of "`member`" attributes. The "`cn`" (*common-name*) attribute of the object contains the short-form name of the object.

4.3.3 NetCache Configuration

The appropriate options are found in the GUI in this section:

- Setup | Authentication | LDAP | General <TAB>

The options are named:

- User ID (UID) Field in the LDAP Directory
- Group ID (GID) Field in the LDAP Directory
- "GroupOfUniqueNames" Field in the LDAP Directory

- "GroupOfUniqueNames" Member Field in the LDAP Directory

Below is an example CLI configuration for operating with Active Directory:

```
config.auth.ldap.uid = samAccountName
config.auth.ldap.gid = memberOf

config.auth.ldap.ugid = cn
config.auth.ldap.member = member
```

Note that common alternative attribute names are `uid` (instead of `samAccountName`), `gid` (instead of `memberOf`), and `uniqueMember` (instead of `member`).

NetCache performs the following operations, in sequence, to search LDAP directories:

- Bind Operation
  - The configured "Bind DN" credentials are used

- Search Operation
  - Search for user objects in the directory; request group memberships if matching object is found

Search Parameters	**Generic**	**Active Directory**
Filter	(<uid> = username)	(samAccountName=username)
Attribute	<gid>	memberOf

- Bind Operation
  - The DN of the user, returned in the previous search, is used
  - Verifies if the supplied password is valid

- Bind Operation
  - The configured "Bind DN" credentials are used

- Search Operation
  - Search for GroupOfUniqueNames objects; request the name of any group objects where the user is found to be a member

Search Parameters	**Generic**	**Active Directory**
Filter	(<member> = DN of user)	(member=DN of user)
Attribute	<ugid>	cn

4.4 Handling Expired Passwords

A response to a client LDAP request includes a result code and an error message. The possible result codes are standardized. The error messages, however, are not: they are vendor-dependent text strings.

The “invalid credentials” result code for a failed bind operation does not provide the client with any information regarding the reason for failure. The two main reasons are:

- Incorrect password for the account
- The account’s password is correct but has expired

A number of vendors provide a diagnostic message indicating the expired password condition in the error message field. NetCache 5.6.2 recognizes the expired password messages for Novell’s directory service and Netscape/iPlanet LDAP services. Future NetCache versions will recognize the message for Active Directory.

When an expired password condition is detected, NetCache has the ability to redirect the end user to a new URL. This gives the capability of describing the situation to the end user rather than simply providing a generic “access denied” message.

The appropriate option is found in the GUI in this section:

- Setup | Authentication | General

The option is named:

- Expired Password URL

An example CLI configuration is:

```
config.auth.expire_url = http://www.x.com/expired_message.html
```

Appendix A: Log Files

The NetCache log files are customizable and can include a wide variety of data. This section discusses just some of the data that may be of interest.

NetCache can log the IP addresses of the peers with which it communicates:

- The “c-ip” field displays the client IP address
- The “s-ip” field displays the IP address to which the client sent its request
- The “r-ip” field displays the IP address of the server NetCache used to satisfy the client request

In this deployment, logging the fields “c-ip s-ip r-ip” would display data as follows:

```
<Client-IP> <NetCache-Accelerator-IP> <Web-server-IP>
```

A field is not currently available for logging the port to which the client connected on NetCache. However, this information can be logged by use of the ACL “log-info” action: the value from an ACL variable can be stored and logged using the “x-acl-info” field. This is shown in the last example configuration at the end of this section.

When a request is made by a client to a reverse proxy, the client believes it is communicating with the source of the content. A relative URL is therefore provided by the client (e.g., /path/file), as opposed to an absolute URL, which would be provided in the case of a forward proxy (e.g., http://www.x.com/path/file).

Logging the “x-request-line” field would display data as follows:

```
“GET /path/file HTTP/1.1”
```

Logging the fields “cs-method x-uri-path x-protocol” would display data as follows:

```
GET /path/file HTTP/1.1
```

Note that the “x-request-line” field is logged with quotes surrounding the values. The two methods above provide similar logging because this is a reverse proxy deployment. For a forward proxy deployment, where the client is aware it is communicating with a proxy, this would not be the case: the “x-request-line” field would contain an absolute URL.

The field “cs-uri” is used to log the absolute URL that a client is requesting. For a reverse proxy deployment, NetCache constructs a URL that includes some information that is used for the server-side connection. This is somewhat unintuitive and is not recommended for a reverse proxy deployment. Data would be displayed as follows:

```
https://<netcache-hostname>:<server-port>/path/file
```

For authentication, the “x-authmethod” field logs the method used to validate the supplied credentials. The “x-domain” field logs, for the case of LDAP, the LDAP server name (or IP address) and port number that was used. The “x-username” field logs the username of the user.

Logging the fields “x-authmethod x-domain x-username” would display data as follows:

```
ldap <ldap-server-ip>:<ldap-server-port> <username>
```

The appropriate options for configuring log files are found in the GUI in this section:

- Setup | HTTP | Logging

The options are named:

- Web Access Log Enable
- Log Format

An example CLI configuration is:

```
config.http.logs.access.enable = on

config.http.logs.access.format. = \\
x-localtime c-ip s-ip r-ip x-hiercode x-transaction x-note
rs(Content-Type) bytes cs-method x-uri-path x-protocol x-
authmethod x-domain x-username
\\
```

Use of the ACL log-info action to capture the NetCache port number is shown below:

```
config.https.acl = \\
auth any
```

```
log-info (cache-ip ‘:’ cache-port)
\\

config.http.logs.access.format. = \\
x-localtime c-ip x-acl-info r-ip x-hiercode x-transaction x-
note rs(Content-Type) bytes cs-method x-uri-path x-protocol
x-authmethod x-domain x-username
\\
```

Appendix B: Access Control Lists

Key elements controlled by the ACLs in this deployment are:

- Allow access only if reverse proxy requests are being made
- Allow certain clients unrestricted access
- Allow all clients access to certain URLs
- Require authentication for all other access
- Insert username header
- Allow certain groups access to certain URLs
- Deny all access

To match URLs, the “url matches”, “url contains”, “url contains-case”, and “url-prefix” keywords can be used. As discussed in the previous section, the internal format of the absolute URL for a reverse proxy deployment can be unintuitive. The best approach may be to use the “url” variable to match on the pathname and filename components only, and to use the “cache-port” and “cache-ip” variables to match on the network destination components. (The latter is not necessary, of course, if NetCache is accepting connections on a single IP address with a single port number.)

An example CLI configuration is:

```
config.https.acl = \\
deny not accel
allow client-ip ip-addr-range
allow cache-port 443 and url contains “/public_path/file”
auth any
user-header any
allow group “cn=testgroup” and url contains “/path/file”
deny any
\\
```

Appendix C: LDAP over SSL

LDAP communication between NetCache and an LDAP service—such as Active Directory—may be protected using SSL. An SSL context needs to be created for connections initiated by NetCache (outbound) and then assigned appropriately.

The appropriate options are found in the GUI in this section:

- Setup | Authentication | LDAP | Secure LDAP <TAB>

The options are named:

- Secure LDAP Enable
- SSL Context Used by Secure LDAP Connections

An example CLI configuration is:

```
config.ssl.contexts = \\
app1_inbound on certfile.pem keyfile.pem inbound off
ssl2,ssl3 all default 5 300 default
secure_ldap on none none outbound on ssl2,ssl3 all default 5
300 default
\\

config.auth.ldap.enable = on

config.auth.ldap.ssl.enable = on
config.auth.ldap.ssl.context = secure_ldap

config.auth.ldap.servers = \\
ldap-server-ip:636
\\

...
```

References

NetCache Deployment

[TR-3071]
Niall Doherty, July 2000
Application of Distributed Web Site Acceleration: Mars Polar Lander
http://www.netapp.com/tech_library/3071.html

[TR-3309]
Niall Doherty, September 2004
Packet Flows in Common Cache Deployments
http://www.netapp.com/tech_library/3309.html

HTTP

[RFC-2616]
R. Fielding et al., June 1999
Hypertext Transfer Protocol—HTTP/1.1
http://www.ietf.org/rfc/rfc2616.txt

[RFC-2617]
J. Franks et al., June 1999
HTTP Authentication: Basic and Digest Access Authentication
http://www.ietf.org/rfc/rfc2617.txt

Authentication

[TR-3336]
Niall Doherty, September 2004
NetCache and Authentication
http://www.netapp.com/tech_library/3336.html

[RFC-1777]
W. Yeong et al., March 1995
Lightweight Directory Access Protocol
http://www.ietf.org/rfc/rfc1777.txt

[RFC-2251]
M. Wahl et al., December 1997
Lightweight Directory Access Protocol (V3)
http://www.ietf.org/rfc/rfc2251.txt

NetApp

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com