NetCache® and Authentication

by Niall Doherty, Network Appliance, Inc.

September, 2004 | TR 3336

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.

Table of Contents

Abstract

This document examines authentication interactions with the NetCache appliance in forward proxy configurations. Authentication in a Microsoft® Windows® environment is discussed in some detail.

1. Introduction

Proxy servers go beyond the packet filtering and inspection provided by firewall devices and offer additional functionality at the application level. A proxy server that provides caching functionality is also known as a "cache server": the two terms will be used interchangeably in this document.

A common use of a proxy server deployed in an enterprise environment—in a forward proxy configuration—is to act as a "gateway" to the Internet and to control and monitor user activity.

The NetCache product from Network Appliance, Inc. is a cache server that has traditionally been used to improve bandwidth utilization and response times by performing caching of common Internet objects. NetCache also supports sophisticated authentication and access control functionality, and this forms the basis of a number of services provided in conjunction with some of our partners, e.g.:

- Access control based on users and groups
- URL filtering and virus scanning
- Logging and reporting on user activity

To provide user activity reporting, and to combine user and group names in conjunction with ACLs for more fine-grained control, it is necessary to perform authentication to determine and verify a user's identity.

This document describes the authentication technologies relevant to NetCache and discusses how they apply to NetCache deployments. Authentication in a Microsoft Windows environment is of particular interest to many customers, and this topic has been discussed in some detail.

The goal of the document is to provide an understanding of the implications of a particular deployment configuration for authentication. This is not a troubleshooting guide; analysis of unexpected behavior is not covered.

A number of topics have been omitted, or only touched upon lightly, and may be covered in a later version of this document, for example:

- Interaction with authentication requests from Web servers
- Authentication in a hierarchical NetCache deployment
- The LDAP and RADIUS protocols
- NetCache joining a Windows domain
- Integration with Netegrity SiteMinder and "cookie authentication"

2. Microsoft Windows Authentication

2.1. Network Authentication Protocols

In a Windows 2000 domain, there are two choices for network authentication: Kerberos V5 [KERB] and Windows NT® LAN Manager (NTLM) [NTLM].

Both of these authentication protocols are based on *shared secrets*. If only two parties know a particular secret, then either party can verify the identity of the other by confirming that the other party knows the secret. Encryption is employed so that the secret itself is not sent across the network.

NTLM was the default authentication protocol for Windows NT 4.0. Kerberos was introduced with Windows 2000, and it is preferred over NTLM. Authentication with older Windows operating systems—as well as certain operations in Windows 2000 domains—uses NTLM.

Kerberos is a more flexible, efficient, and secure authentication protocol. A key difference is that an application service can verify a client's identity without having to contact any other system.

2.2. Account Databases

User account databases are managed by centralized servers known as domain controllers. Windows processes refer to users not by their username, but by their Security ID (SID): a SID is a unique identifier assigned to user, group, and computer accounts.

Passwords are not stored in the database: instead, hashes of the passwords are stored. A hash is a "one-way function": it is impossible to derive the password, mathematically, from the hash. The following two hashes are used:

- **LM hash.** This is created using a DES encryption [DES] of an uppercase, ASCII representation of the password.
- **NTLM hash.** This is created using an MD4 hash [MD4] of a mixed-case, Unicode representation of the password.

In a Windows NT domain, the user account database is known as the Security Accounts Manager (SAM) database. Windows 2000 introduced Active Directory (AD): this database of network information incorporates the user account data. Access to this database is provided primarily through LDAP, although other access methods are supported for backward compatibility.

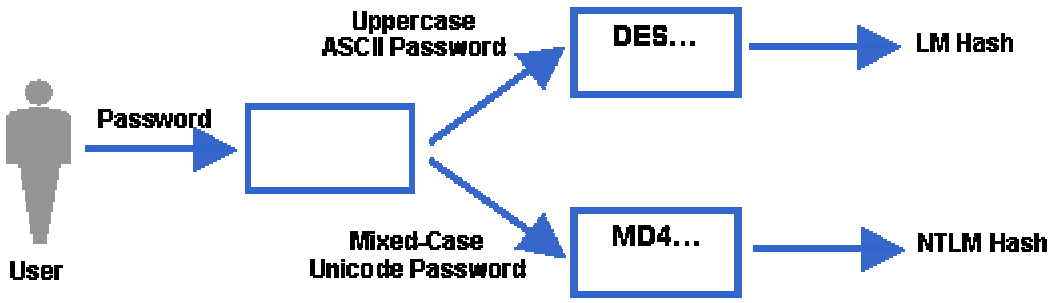

Figure 1) Creation of LM and NTLM hashes.

2.3. Kerberos Overview

In a network using Kerberos authentication, each party has a long-term **secret key**. A trusted *third party*—the **Key Distribution Center** (KDC)—has knowledge of these secret keys.

At the beginning of a communication session, a client sends the username of the authenticating user and an **authenticator** to the server. The authenticator contains the username and a timestamp, both encrypted with a short-term **session key**.

The KDC is responsible for generating these session keys. It creates a copy for both the client and the server and encrypts these copies with their respective secret keys.

Authentication takes place as follows:

1. The client contacts the KDC (1 in figure 2) to request a short-term **session key** for communication with a particular server.
2. The KDC sends two items to the client (2 in figure 2):
    a. The session key, encrypted with the client's secret key.
    b. A **ticket** for the server: this includes—encrypted with the server's secret key—the session key and authorization data. In Windows, the authorization data includes a list of SIDs representing the groups to which the user belongs.
3. The client decrypts its copy of the session key using its secret key. The authenticator is created by encrypting the username and the timestamp with the session key.
4. The client sends the username, authenticator, and ticket to the server (3 in figure 2).
5. The server decrypts its copy of the session key from the ticket using its secret key; it decrypts the authenticator with the session key.
6. Examination of the decrypted authenticator information verifies the identity of the client.

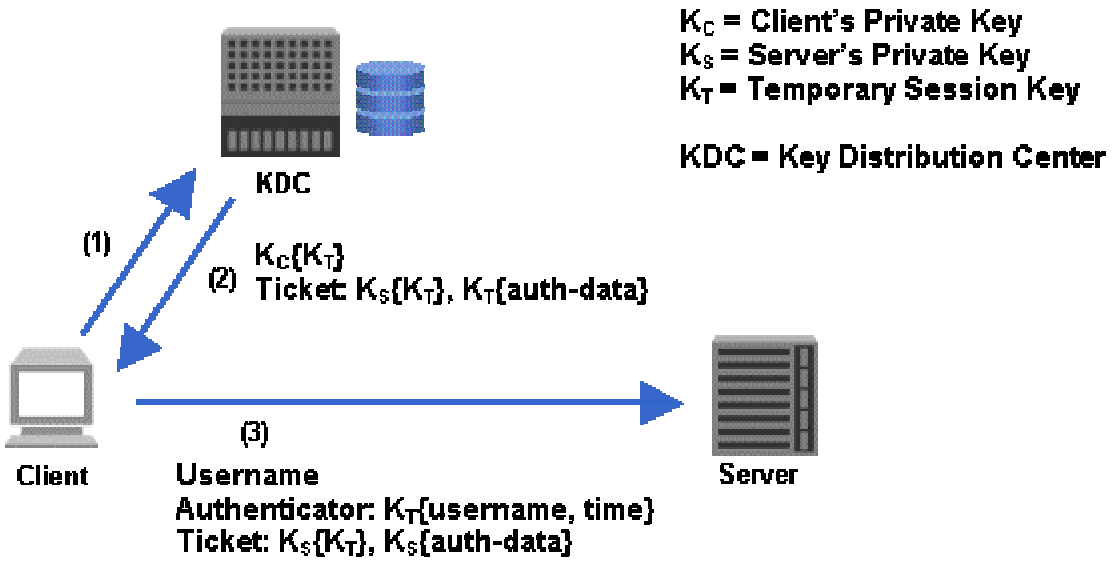

Figure 2) Overview of Kerberos authentication.

In a Windows 2000 domain, the KDC is implemented as a domain service. Both the KDC and Active Directory (AD) services are located on every domain controller (DC).

2.4. NTLM

2.4.1. Overview

NTLM uses a challenge-response mechanism. A challenge—formally known as a nonce—is sent to the client. The client generates a response by cryptographically combining the nonce with the password of the authenticating user and returns this. A domain controller has knowledge of both the user's password and the nonce sent to the client: it can verify a user's identity by generating a response itself and comparing this with the response sent by the client.

Three messages are exchanged during the NTLM authentication process:

- **Message 1.** The client sends a list of supported features and, optionally, the name and domain of the client workstation (not the user).

- **Message 2.** The server responds with a list of the features that will be used and, optionally, information about itself. It also sends a nonce—an eight-byte challenge—that the client uses to generate a response.

- **Message 3.** The client sends the name of the client workstation and the username and domain of the authenticating user. It also sends its response to the challenge.

2.4.2. Client, Application Server, and Domain Controller Interactions

Application servers often provide services that require user authentication. Since only domain controllers have knowledge of user passwords, these application servers need to communicate with a domain controller to verify a user's identity. Two methods can be used:

- **Pass-through.** When a client requests service, the server contacts a DC and receives a nonce, which it sends to the client. The client's response is then sent through by the server to the DC. The DC then replies to the server, stating whether authentication was successful or not.

- **NetLogon service.** To use this method, a server must first become a member of the domain. The server generates the nonce itself: when it receives the response from the client, it sends the nonce along with the necessary information in the client's message to the DC. On successful authentication, the DC returns a list of SIDs to the server. The first of these represents the user; the others represent the groups to which the user belongs.

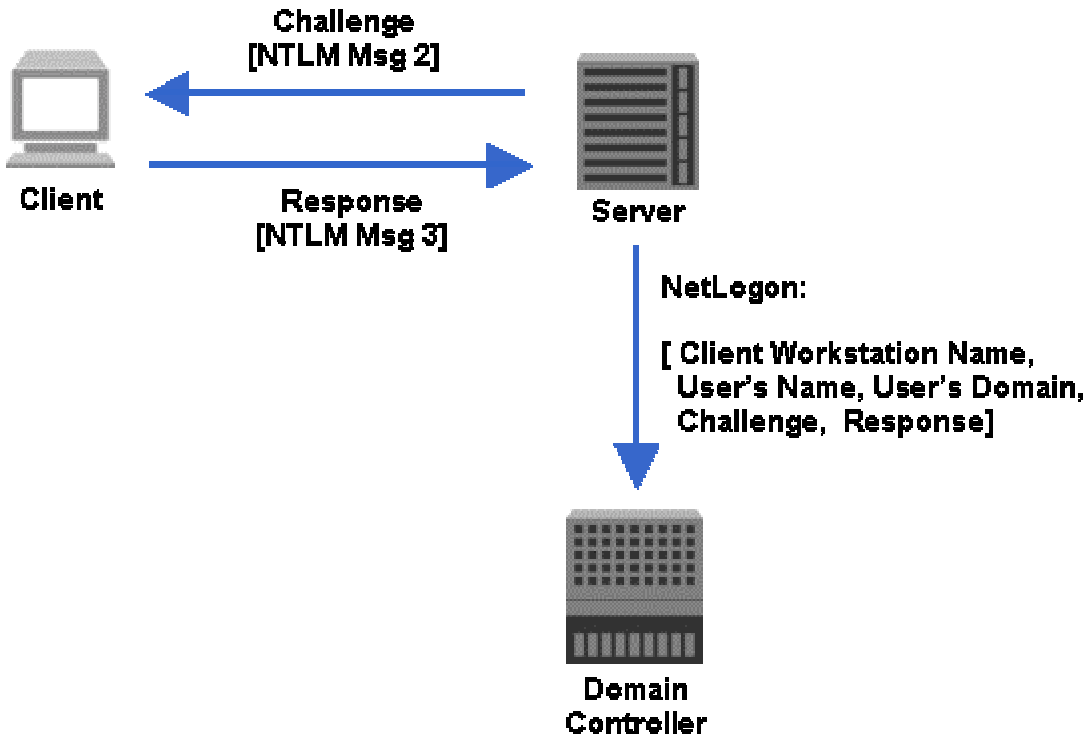

Figure 3) Validation of credentials using the NetLogon method.

Both of the above methods use the Microsoft Server Message Block (SMB) protocol for communication between the server and the DC. The NetLogon service uses SMB transaction messages to make Microsoft Remote Procedure Calls (MS-RPCs) to the DC.

Note: In a Windows 2000 domain, when validating NTLM credentials, an application server will authenticate itself to the DC using Kerberos at the beginning of the SMB communication session. Examination of a packet trace would show Kerberos being used by the server for identification and then the RPC calls for the NTLM credential verification of the client.

2.4.3. NTLM Responses

The third NTLM message has two response fields: the LM/LMv2 and NTLM/NTLMv2 fields. The contents of these fields are controlled by the value of a registry setting: *LMCompatibilityLevel* in Windows 2000 and later and *LMCompatibility* in earlier versions of the operating system.

At the default value in most versions of the operating system—zero—the **LM response** and **NTLM response** are sent by the client. The responses are created as follows:

- **LM response.** This is created using a DES encryption of the **LM hash*** and the server-generated challenge.
- **NTLM response.** This is created using a DES encryption of the **NTLM hash*** and the server-generated challenge.

*Recall that the LM hash and NTLM hash are simply encrypted representations of the user's password.

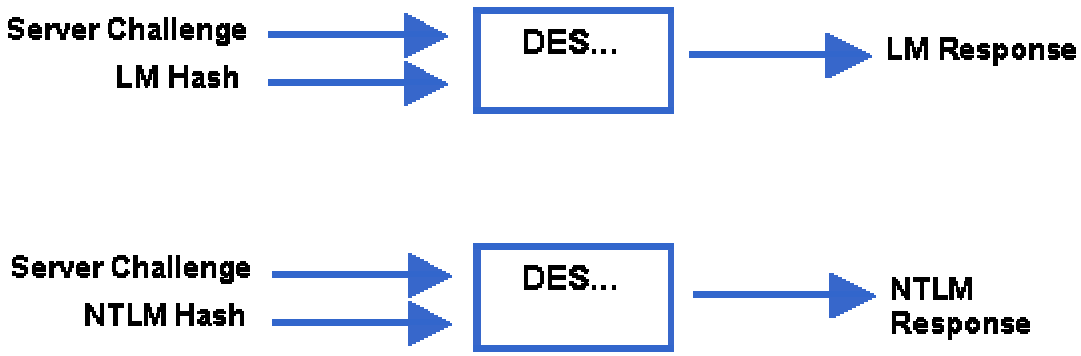

Figure 4) Creation of LM and NTLM responses.

If the registry key is set to three or higher, the LMv2 and NTLMv2 responses are sent by the client. These responses are calculated by combining a newer NTLMv2 hash with data generated by both the client and the server:

- **Blob.** This block of data includes a timestamp and an eight-byte challenge, generated by the client.

- **NTLMv2 hash.** This is created by applying the HMAC-MD5 authentication code algorithm [HMAC] to a concatenation of the user's username and the name of the logon destination, using the original NTLM hash as the key.

The LMv2 and NTLMv2 responses are created as follows:

- **NTLMv2 response.** HMAC-MD5 is applied to a concatenation of the blob and the server challenge, using the NTLMv2 hash as the key. The response is a concatenation of the HMAC-MD5 result and the blob.

- **LMv2 response.** This is used for pass-through compatibility with older servers. It is created in a similar fashion to the NTLMv2 response: the difference is that only the client-generated challenge is used instead of the entire blob.

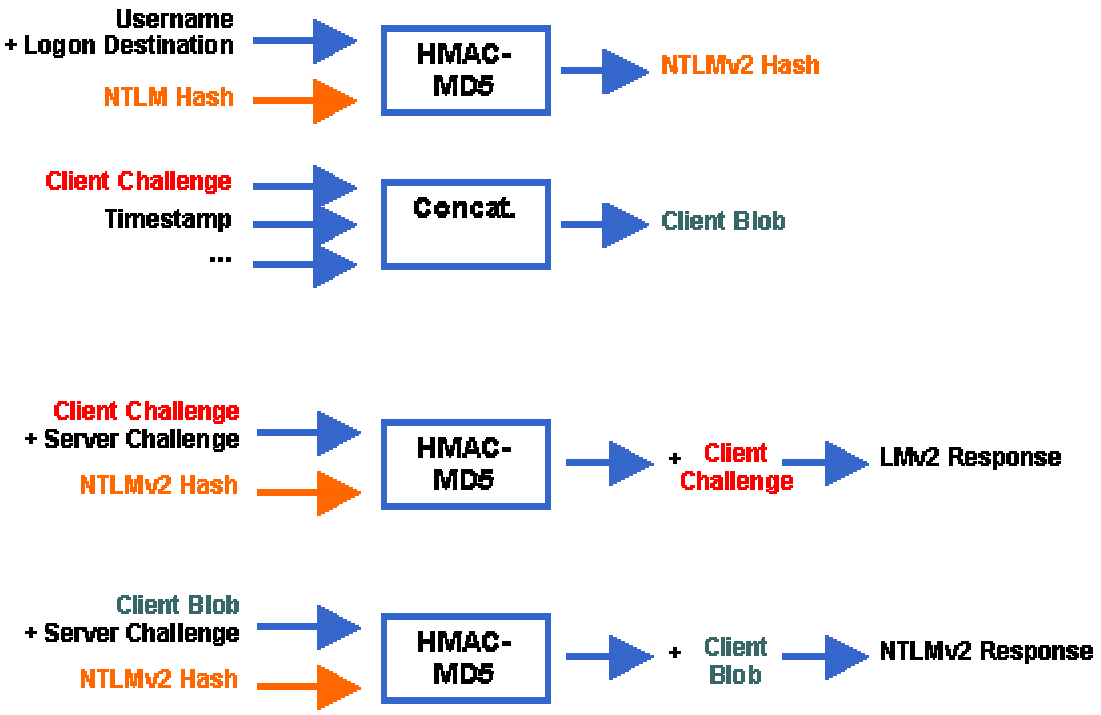

Figure 5) Creation of LMv2 and NTLMv2 responses.

The predominant form of NTLM authentication—NTLMv1—in use at the time of writing (August 2004) always generates the same response for a given nonce. Simply by changing the value of a registry key, however, clients will favor the stronger algorithms introduced with NTLMv2. A key difference is that—because of the client-generated data—responses for a given server-generated nonce are likely to be different each time.

Additionally, if the earlier LM and NTLM algorithms are used, newer versions of the operating system can negotiate "Session Security." In this case, the client—similar to NTLMv2—generates data that is used in the creation of the response. Again, the responses for a given server-generated nonce are likely to differ.

To summarize, NTLMv2 and NTLMv1 with Session Security both provide different responses to the same server-generated nonce. An example of how this affects server applications is described in section 3.4.2.

3. NetCache and HTTP Proxy Authentication

3.1. HTTP Authentication Overview

There are two types of access authentication in HTTP [RFC-2616]:

- **Proxy authentication.** Clients present credentials to a proxy, in order to gain access to resources on Web sites beyond that proxy.

- **WWW authentication.** Clients present credentials to a Web site, in order to gain access to a resource on that Web site.

This document is primarily concerned with the former. The following items will be discussed:

- Obtaining the user's credentials by the browser
- Sending of the credentials from the browser to NetCache
- Validation of the credentials by NetCache

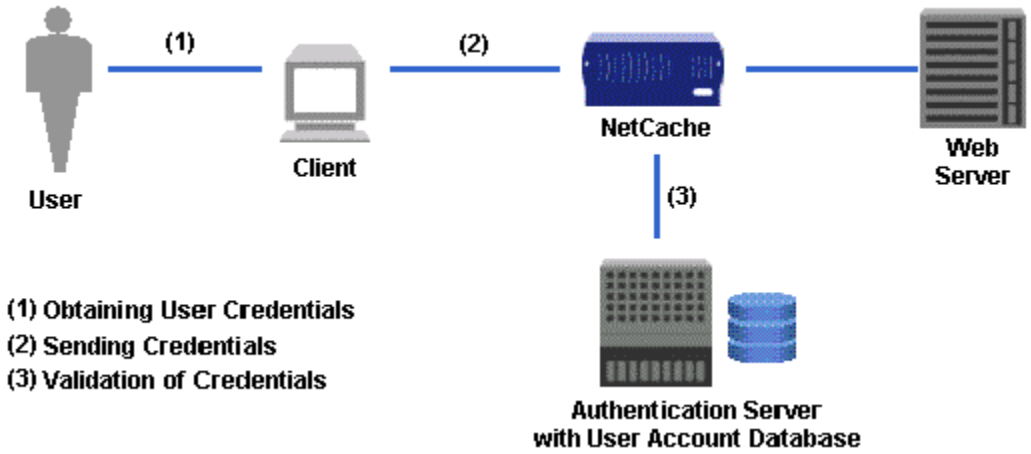

Figure 6) Proxy authentication and NetCache.

3.2. Sending Credentials

3.2.1. HTTP Basic Authentication

The most commonly implemented method of sending credentials from a browser is known as "Basic authentication" [RFC-2617].

The browser concatenates the username, a colon, and the plain-text password and then base-64 encodes the result. The result is sent in an HTTP request header.

Note that base-64 encoding is not encryption: the plain-text password can be derived trivially with a simple operation.

With HTTP Basic, authentication is on a **per-request basis**. Each request needs to be authenticated separately, if required. This follows the basic principles of the HTTP design, where state information is not maintained for connections or requests:

- Each HTTP request is considered without reference to the TCP connection on which it arrived.
- Each request is considered without reference to any requests that may have preceded it.

3.2.2. Windows Network Authentication Protocols

Internet Explorer introduced support for the use of both NTLM and Kerberos over HTTP for sending credentials as an alternative to HTTP Basic. Other clients now also include support, e.g., Mozilla-based browsers and Yahoo! Messenger.

In Windows environments, authentication is typically performed at the beginning of a communication session, and all requests made during that session are considered authenticated. NTLM and Kerberos over HTTP follow this philosophy: authentication is **connection-oriented**. The first TCP connection—where the server indicates authentication is required—is shut down, and a new one is initiated. This new TCP connection needs to be a private, persistent connection, using either the "Connection: Keep-Alive" header for HTTP/1.0 or the standard persistent connections of HTTP/1.1. Authentication occurs at the beginning of this second HTTP connection. All further HTTP requests on that connection are considered authenticated and are not challenged for authentication.

The benefit of these two methods—compared to HTTP Basic—is that the password is not transmitted in clear text.

3.3. Obtaining User Credentials

Browsers obtain credentials for HTTP Basic authentication by prompting the user: a two-field dialog box is displayed, allowing the user to enter a username and password. Browsers store the credentials in memory if authentication is successful. When the user accesses that service again (typically, on every request in the case of a proxy), the browser automatically includes the credentials previously entered with the request. Thus, users are normally only prompted for authentication once per browser session for each service (Web site or proxy) that is accessed.

Windows stores the username and (encrypted) password of a user logged onto a client, and this provides the capability for Internet Explorer to send credentials without prompting the user. This is known as "automatic logon," "promptless authentication," or "Single Sign-On" (SSO) and is used for NTLM and Kerberos authentication.

Internet Explorer supports the notion of "Web content zones" and applies different security policies to servers in different zones. Typically, Web sites and proxy servers in an organization's intranet are considered to be in a "trusted" zone.

When NTLM or Kerberos authentication is being performed, Internet Explorer will not perform an automatic logon in the following cases:

- The user is not logged into a domain.
- The proxy server or Web site is not in a "trusted" Web content zone.

The user is presented, instead, with a three-field dialog box; a field is now available for the domain of the user. The transfer of credentials is still secured through the normal NTLM or Kerberos methods; only the automatic logon process is not used.

Since proxy servers are generally in the intranet, an automatic logon is normally performed for proxy authentication. For Web sites outside the corporate network requesting Web site (origin server) authentication with NTLM or Kerberos, an automatic logon is generally not performed; the user is prompted instead.

Non-Internet Explorer clients may not be able to perform an automatic logon for NTLM or

Kerberos: the end user needs to provide the name, password, and domain either with a configuration option in the client or through a dialog box.

3.4. Validation of Credentials

3.4.1. Validation Methods

Table 1 lists the various ways NetCache can validate the credentials it receives.

Credentials Received Using...	Can Be Validated...
HTTP Basic	• With a local NetCache database • With a RADIUS server • With an LDAP server • With a DC using the NetLogon service(1)
NTLM over HTTP	• With a DC using the NetLogon service(2)
Kerberos over HTTP	• Internally, by examination of the ticket(2)

Table 1) Credential validation by NetCache.

Note 1: NetCache can validate HTTP Basic credentials with a domain controller. In this scenario, NetCache needs to supply some data that is normally provided by the client:

- **Domain.** A client cannot send domain information with HTTP Basic authentication. NetCache uses, instead, the domain that it has joined.

- **NTLM response.** NetCache can generate the response itself, since it has knowledge of the username and password of the authenticating user, as well as the algorithms necessary to compute the response.

Although the password is sent to NetCache unencrypted, the centralized Windows user account database can still be used, which offers a certain convenience. The following are examples of when this scenario may occur:

- If a client does not support NTLM, credentials can be sent using HTTP Basic authentication (e.g., old Netscape® browsers).

- If validation of credentials received by NTLM or Kerberos fails, NetCache can challenge for HTTP Basic authentication.

Note 2: Credentials received via NTLM or Kerberos over HTTP cannot be validated using any other service, such as RADIUS or LDAP. These services would require that NetCache had access to the user's password, which it does not.

Each method of validation protects the password. NetCache sends an MD5 hash [MD5] of the password to the authentication server when using RADIUS [RFC-2138]. LDAP [RFC-1777] and [RFC-2251] allows negotiation of the encryption method, commonly the SHA-1 algorithm [SHA-1]; in addition, the entire LDAP session can be encrypted using SSL. NTLM involves the sending of challenges and encrypted responses to a domain controller for validation. Finally, Kerberos does not require NetCache to contact any authentication server for validation.

3.4.2. Reducing Load on Authentication Servers

Browsers can potentially send credentials with every HTTP request made. Validating credentials for every request puts significant strain on authentication servers. NetCache uses the techniques described below to alleviate the load on authentication servers.

For credentials received via HTTP Basic, NetCache caches the username and password for a certain period (default is one hour) if the validation result is successful. Further requests with matching credentials can be immediately validated without needing to contact the authentication server.

NTLM credentials do not contain any clear-text password and have the additional property of changing each time a new nonce is provided to the client. However, NetCache has the ability to "cache" NTLM credentials (prior to NTLMv2) by ensuring that each user always receives the same nonce (unique to that user). If this capability is used, NetCache can validate credentials of users who have already successfully authenticated by checking its internal NTLM credential cache (each entry contains a username, nonce, and expected response).

The session security used for NTLMv1 and the newer algorithms introduced by NTLMv2 cause clients to generate different responses to the same nonce. A modified technique is, therefore, required. NetCache sets a cookie for each user who successfully authenticates. This cookie, arriving in an HTTP request, indicates to NetCache that the user at that client has recently authenticated successfully. The cookie is designed so that it is difficult for others to reuse: it includes the client IP address, a timestamp, and a TTL. This information is all encrypted using the Triple DES algorithm. Since cookies can be set only on a per-domain basis, the result is that validation requests need to be made to the domain controller once per user, for each domain that that user accesses (e.g., once for .yahoo.com, once for .ebay.com, etc.).

Finally, as described earlier, credentials received via Kerberos are always validated without the need to contact any authentication server.

3.5. Group Memberships

In most environments, users normally belong to one or more groups. NetCache access control lists (ACLs) allow policies to be applied to groups of users, as well as individual users. The username is provided by a client when authentication is attempted. The groups a user belongs to are found in a variety of ways, depending on what technology is being used. A user's group memberships are also included in his/her entry in the server-side credential cache discussed in the previous section.

A successful authentication with an LDAP server results in a list of group memberships being returned.

The RADIUS protocol does not have any notion of groups of users. However, NetCache can be configured to interpret the values in a specific RADIUS response attribute as representing group names. Use of this capability requires that the RADIUS server be configured appropriately.

When NTLM credentials are verified with a domain controller, NetCache receives a list of SIDs representing the groups in which the user is a member. Lookups to the domain controller are required to convert the SIDs to group names. NetCache maintains a table (cache) mapping SIDs to group names: typically, many lookups will be made soon after NetCache boots; the number of lookups will die down gradually as the table becomes populated.

In a Windows environment, the Kerberos ticket sent by the client contains a list of SIDs. While the

user identity can be validated simply by examination of the ticket, queries to a domain controller still need to be made to convert the SIDs to group names, as for the NTLM case above.

4. Deployment Considerations

4.1. Forward Proxy Configurations

In forward proxy mode, a client makes a request that is explicitly targeted at an origin server. The cache is deployed as a "middleware" device: it, instead of the server, handles the request for the client.

4.1.1. Transparent Redirection

Transparent redirection refers to a configuration whereby a networking element located in the path of the client-server traffic flow intercepts all—or some portion of—that traffic and sends it to another device—in this case, a cache.

In this scenario, the client is unaware of the existence of the cache and believes it is sending packets directly to the server.

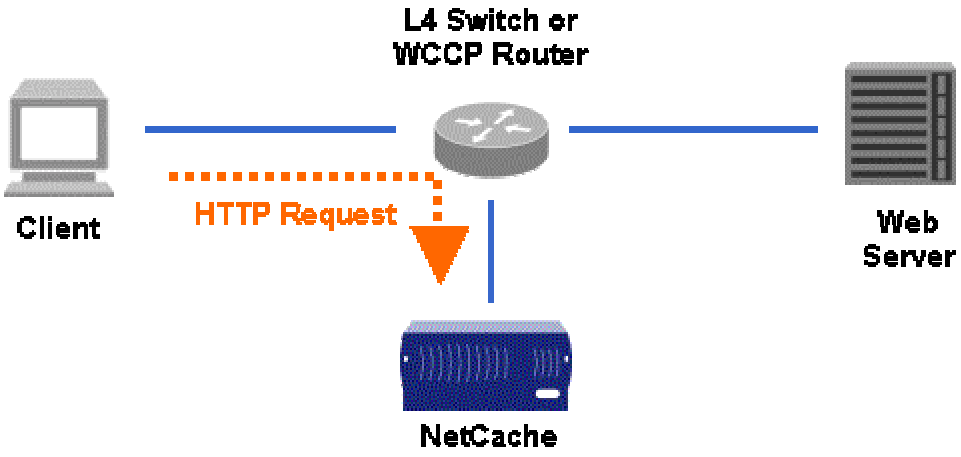

Figure 7) Transparent redirection.

The advantage of this solution is that no changes to the client are required. The disadvantage is that modifications to the network infrastructure must be made: either a "layer 4 switch"—from vendors such as Foundry and Alteon—needs to be inserted at an appropriate point or a Cisco device supporting WCCP (Web Cache Control Protocol) needs to be used.

4.1.2. Browser Configuration

It is possible to configure all common Web and streaming clients (and many FTP clients) to use a proxy-either by directly naming the proxy or by using a JavaScript automatic proxy configuration script (also known as a "proxy.pac" file).

In this scenario, the client communicates directly with the cache. Unlike a transparent redirection

solution, no network changes need to be made. The following methods can be used to make the client aware of the cache:

- Direct configuration of the client by the end user
- Remote configuration of the client by the IT organization, e.g., using the Internet Explorer Admin Kit
- Automated discovery of the automatic proxy configuration script through WPAD (Web Proxy Auto-Discovery)

Note: NetApp Technical Report 3309 [TR-3309] discusses NetCache network deployments in detail.

4.2. Transparent Redirection and Authentication

4.2.1. The Problem with Proxy Authentication

The main challenge is that current browser versions will not respond to a request for proxy authentication if they are unaware of the presence of a proxy server.

NetCache attempts to work around this browser limitation in a transparent redirection deployment by using the WWW authentication method instead. To a browser, it appears that the *Web sites* are requesting authentication. NetCache ensures, of course, that the actual Web sites do not see any credentials sent by the browsers.

4.2.2. Internet Explorer Automatic Logon

In a transparent redirection environment, Internet Explorer will receive authentication requests from what *appear* to be Internet Web sites. As explained in section 3.3, it is likely that an automatic logon will not be performed in this case: i.e., users will be prompted for their credentials.

4.2.3. NetCache Client-Side Authentication Cache

Because different Web sites appear to be requesting authentication-instead of a single proxy server—browsers will not send the existing credentials when accessing new sites. This implies that users would be constantly prompted for their credentials.

NetCache attempts to solve this problem by not requesting authentication from clients that have recently sent credentials that were successfully validated. NetCache implements a cache based on the client IP address. Credentials are not requested from clients again until a certain period—a configurable time-to-live (TTL) value—has expired.

Significantly, the TTL for a particular IP address is reset each time a request is received from that IP address. Authentication is only required again if there is no activity from that IP address for a duration of time equal to the TTL. It behaves as an idle, or inactivity, timer.

This solution is clearly based on the assumption that there is one user per IP address. There are some scenarios where this may not be the case.

- **Multitasking operating systems.** It is possible for multiple users to run a browser on a particular computer with a multitasking operating system, displaying the GUI locally (using, say, XWindows). This particular scenario is the least likely cause of concern.

- **Accessing another user's computer.** A user may logon to a computer that was recently used by another user. Assuming users lock their screen when away from their desk, this can only occur when they have logged off. In this latter case, the risk can be mitigated by shortening the TTL of the client-side authentication cache.

- **Network Address Translation (NAT).** In an enterprise environment, this is typically not a problem. If it does arise, however, it is most likely a result of an NAT device at another geographical location. In this case, there is a possible solution: a NetCache appliance could be deployed at that location, behind the NAT device. NetCache can be configured in such a way that the (trusted) "child" NetCache appliance passes the client IP address to the "parent" NetCache appliance using an HTTP request header (X-Forwarded-For). The parent can extract this IP address for use by its internal client-side authentication cache.

4.3. Multiple Netcache Appliances and Authentication

When traffic is distributed across multiple NetCache appliances, there are a number of methods that can be used to load-balance the traffic, e.g., round-robin, least existing connections, and hash on various parameters (source or destination IP addresses and port numbers, URL, etc).

A common method is to hash on the destination (server) IP address of the packets sent by the client: this gives an even distribution of traffic and reduces to a minimum any duplication of content across the NetCache appliances.

To avoid users needing to authenticate to each NetCache appliance, it is necessary to ensure that each client communicates with a single NetCache appliance. Load balancing based on a hash of the source (client) IP address achieves this goal. The main disadvantage is that content is duplicated across the NetCache appliances. This is generally not a major concern, since the disk space required for good performance is generally less than that present on most HTTP caches today.

Each of the following scenarios supports the required load-balancing method:

- In a transparent redirection deployment, both layer 4 switches and WCCP routers support use of the hash method.

- In a browser configuration deployment, where the browser communicates with a virtual IP address (VIP) on a server load balancer (SLB) device, that device can usually also load-balance using the hash method.

- In a browser configuration deployment using a proxy.pac file where the client communicates directly with multiple servers, it is possible to implement similar functionality in JavaScript.

4.4. Kerberos Deployments

Kerberos credentials can only be validated by the server intended to receive those credentials: the ticket is encrypted with the server's secret key.

This implies that Kerberos authentication cannot work in deployments where a client is communicating with a server other than that which was intended.

For example, Kerberos authentication cannot be used in the following deployments:

- A transparent redirection deployment, where the client believes it is communicating with the Web server and not a proxy server
- A browser configuration deployment, where the browser is directed to a VIP on an SLB device: the client does not know with which proxy server it is actually communicating

5. Conclusions

In deployments where the browser is aware of a proxy server and communicates with it directly, users are generally prompted for a name and password once per browser session.

Using Microsoft network authentication protocols over HTTP can provide the following benefits:

- An automatic logon can be performed (users are not prompted for a password).
- The password is not sent across the network.

Authentication can be made to work with NetCache in a transparent redirection deployment, with the following limitations:

- The automatic logon process referred to above will not be used by the browser.
    - Users are prompted, as for HTTP Basic.
- Browsers will prompt more frequently for user credentials.
    - Prompting is controlled by a configuration option (an inactivity timer).
- There is a small risk of a user being identified as another, for the purpose of Web access control.

If a choice of deployment configuration is available, the following should be considered:

- Transparent redirection:
    - Network configuration changes are required.
    - Client modifications are not required.
    - ACLs and URL filtering work well.
    - There are limitations when authentication is used.
- Browser configuration:
    - Network configuration changes are not required.
    - Client modifications are required.
    - ACLs, URL filtering, and authentication work well.

Finally, realize that when authentication is introduced, NetCache depends—except in the case of Kerberos—upon the responsiveness and availability of authentication servers for successful operation.

Appendix A: Client Authentication Protocols

HTTP Basic

The initial proxy authentication exchange begins with the client making a normal HTTP request:

```
GET http://www.x.com/ HTTP/1.0
```

NetCache then responds, indicating that Basic authentication is required:

```
HTTP/1.0 407 Proxy Authentication Required
Proxy-Authenticate: Basic realm="NETCACHE"
```

Next, the client makes the request again, this time including the credentials:

```
GET http://www.x.com/ HTTP/1.0
Proxy-Authorization: Basic YWRtaW46TmV0Q2FjaGU=
```

If the credentials are invalid, another 407 response is returned, and the client can retry.

If the credentials are valid, a successful response is returned:

```
HTTP/1.0 200 OK
```

A successful initial exchange generates two HTTP log file entries:

1. Status code 407, proxy authentication required (Access Denied)
2. Status code 200, a successful response

NTLM Over HTTP

As with HTTP Basic authentication, an initial proxy authentication exchange begins with the client making a normal HTTP request:

```
GET http://www.x.com/ HTTP/1.0              .....1st HTTP Request
```

NetCache then responds, indicating that NTLM is the preferred method of authentication:

```
HTTP/1.1 407 Proxy Authentication Required
Proxy-Authenticate: NTLM
Proxy-Authenticate: Basic "realm=NetCache"
```

Next, the client—using a new HTTP connection—-makes the request again, embedding the first NTLM message within the Proxy-Authorization header:

```
GET http://www.x.com/ HTTP/1.0                .....2nd HTTP Request
Proxy-Authorization: NTLM ABCDEFG...          .....1st NTLM Message
Proxy-Connection: Keep-Alive
```

The second stage in the NTLM negotiation now takes place, with NetCache sending the challenge to the client in the Proxy-Authenticate header:

```
HTTP/1.0 407 Proxy Authentication Required
Proxy-Authenticate: NTLM HIJKLM...            .....2nd NTLM Message
```

The client sends the NTLM response—completing the sending of credentials—with the third HTTP request:

```
GET http://www.x.com/ HTTP/1.0                .....3rd HTTP Request
Proxy-Authorization: NTLM NOPQRS...           .....3rd NTLM Message
Proxy-Connection: Keep-Alive
```

If the credentials are valid, a successful response is returned, as in the case of HTTP Basic.

A successful initial exchange generates three HTTP log file entries:

1. Status code 407, proxy authentication required (Access Denied)
2. Status code 407, proxy authentication required (Partial Authentication)
3. Status code 200, a successful response

Kerberos Over HTTP

As with the previous methods, an initial proxy authentication exchange begins with the client making a normal HTTP request:

```
GET http://www.x.com/ HTTP/1.0
```

NetCache then responds, indicating that the client may perform either NTLM or Kerberos authentication:

```
HTTP/1.1 407 Proxy Authentication Required
Proxy-Authenticate: Negotiate
Proxy-Authenticate: Basic realm="NETCACHE"
```

If the client cannot perform Kerberos, it will perform NTLM instead, proceeding as in the previous example with the second HTTP request. Note, however, that the term "Negotiate" is used—instead of "NTLM"—in the HTTP headers. If, however, Kerberos can be performed, the client sends all necessary information—username, authenticator, and ticket—in the next HTTP request:

```
GET http://www.x.com/ HTTP/1.0
Proxy-Authorization: Negotiate ABCDEFG...
Proxy-Connection: Keep-Alive
```

If the credentials are valid, a successful response is returned, as in the case of HTTP Basic.

A successful initial exchange generates two HTTP log file entries:

1. Status code 407, proxy authentication required (Access Denied)
2. Status code 200, a successful response

Notes:

- Internet Explorer (IE) 5.x introduced support for Kerberos. Unfortunately, the ability to authenticate using Kerberos with a **proxy server** was removed in IE 6.0.
- For both NTLM and Kerberos, the connection used for the initial request is shut down—if it is a persistent connection—after NetCache has responded indicating authentication is required. The next connection needs to be a persistent connection. Authentication is performed only for the first HTTP request on that connection: subsequent requests on the same HTTP connection do not need to be authenticated, and log file entries would show only 200 status codes for these.

Proxy Authentication Versus Web Server Authentication

The previous three sections focused on authentication with a proxy server. Authentication with a Web server is a similar process. The key differences are:

- The 401 status code is used instead of the 407 status code.
- The "WWW-Authenticate" header is used instead of the "Proxy-Authenticate" header.
- The "Authorization" header is used instead of the "Proxy-Authorization" header.

Appendix B: NetCache Feature Releases

Table 2 shows when the various authentication features were introduced for NetCache.

Release	FCS Date	Functionality Introduced
NetCache 3.4	07/1999	LDAP
NetCache 4.0	09/1999	RADIUS
NetCache 5.0	10/2000	Microsoft NTLM
NetCache 5.2	08/2001	Microsoft Kerberos
NetCache 5.5	06/2003	Secure LDAP (LDAP over SSL) Netegrity SiteMinder integration/cookie authentication
NetCache 5.6	01/2004	RADIUS: Attribute-based group support
NetCache 6.0	12/2004	Microsoft NTLM: NTLMv2 support

Table 2) Introduction of NetCache authentication features.

References

NetCache Deployment

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

Microsoft Network Authentication

[NTLM]
Eric Glass, 2003
The NTLM Authentication Protocol
http://davenport.sourceforge.net/ntlm.html

[KERB]
Microsoft, July 1999
Windows 2000 Kerberos Authentication
http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerberos.asp

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 Protocols

[RFC-2138]
C. Rigney et al., April 1997
Remote Authentication Dial In User Service (RADIUS)
http://www.ietf.org/rfc/rfc2138.txt

[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

Cryptography

[DES]
Federal Information Processing Standards Publication 46-2, December 1993
Data Encryption Standard (DES)
http://www.itl.nist.gov/fipspubs/fip46-2.htm

[MD4]
R. Rivest, April 1992
The MD4 Message-Digest Algorithm
http://www.ietf.org/rfc/rfc1320.txt

[MD5]
R. Rivest, April 1992
The MD5 Message-Digest Algorithm
http://www.ietf.org/rfc/rfc1321.txt

[HMAC]
H. Krawczyk et al., February 1997
HMAC: Keyed-Hashing for Message Authentication
http://www.ietf.org/rfc/rfc2104.txt

[SHA-1]
D. Eastlake and P. Jones, September 2001
US Secure Hash Algorithm 1 (SHA1)
http://www.ietf.org/rfc/rfc3174.txt