Samba 4 - Active Directory

Andrew Bartlett
abartlet@samba.org

5th January 2005

Except where otherwise indicated, this thesis is my own original work.

5th January 2005

To my Grandmother

Acknowledgements

This thesis would not be possible without the work of many, many people. It is traditional to first thank my parents: for it was their support, in so many forms, that started me down the road that has ended in Software Engineering.

For mentoring me into and on the Samba Team, and for giving me the rope I needed, I must give thanks to Dr Andrew Tridgell (tridge) and the entire Samba Team.

To my supervisor, Dr Eric Charles McCreath, for taking on a research project at little notice, and midway though the year.

For his close attention to all the Kerberos-related changes, Love Hörnquist Åstrand.

For reviewing and correcting the raw thesis draft, both technically and grammatically, I give my hearty thanks to Dr Eric Charles McCreath, Vance Lankhaar, Jim McDonough, Bruce Bartlett, Jelmer Vernooij, Luke Howard and Dr Andrew Tridgell.

To the Samba Team, and it's supporters for providing the infrustructure on which this thesis has been developed - this thesis has been developed in public, with a full version control history available from:

http://websvn.samba.org/cgi-bin/viewcvs.cgi/trunk/samba4-ad-thesis/?root=lorikeet

Abstract

It has been said (by Sun Microsystems) that ‘the network is the computer’. While this may take it a little too far, networked systems form the heart of every modern enterprise, and at the heart of every modern network are directories of various sorts, which document and control it.

Samba is many things, but primarily a file and print server, that has for over 10 years emulated the Microsoft’s products in this area. In more recent times, and particularly with Samba 3.0, it has taken on new roles in running networks, as a ’Domain Controller’, compatible with the protocols used in NT4.

Samba version 4 is already a massive leap forward in the way Samba is designed, and built. This thesis attempts to take that further, but examining the protocol basis and implementation details adding support for hosting the Kerberos network authentication system into Samba4’s partial implementation of an Active Directory Domain controller.

Active Directory forms the heart of Microsoft’s modern network architecture, and is the heart of many corporate networks. Producing a compatible product is important, if the Samba project is to remain relevant into the future.

In the process, this thesis describes the authentication problem space, and the existing protocols, in particular Microsoft’s proprietary NTLM and Microsoft’s extensions to Kerberos.

By making these changes to Samba version 4, we have progressed closer to (but not yet succeeded in) creating an implementation compatible with Active Directory.

Contents

Chapter 1

The Problem

The fundamental aim of my research project is to document the addition of Kerberos KDC[1] support to the early implementation of Active Directory Domain Controller compatibility in Samba version 4.0.

This is an area of great collaboration, especially between members of the Samba Team[46], and forms the culmination of over 3 years personal effort as a member of that team. In researching this thesis, I have been working on Samba4 since May 2004, and developed the authentication and GENSEC security subsystems (described in Section 11.2) as well as bringing some of my previous work into the Samba4 framework.

This thesis contributes a solid documentary basis for the work going forward, as well as specific forward development in the Kerberos area.

1.1 Building blocks:

This research is based on a solid foundation, utilizing ideas and implementations found within the following Open Source software packages.

Samba3 A production, NT4 replacement DC (Domain Controller)

Samba4 The new development branch of Samba[28]

Heimdal Kerberos A Kerberos 5 implementation with a pluggable backend system.

This thesis also builds on my previous research project, into a Windows compatible Virtual Private Networking solution[3].

[1] In Kerberos, the KDC (Key Distribution Center) forms the core of the Kerberos authentication system. Users 'log in' with a request to the KDC, and the KDC stores (or has access to) all the passwords.

1.2 Samba4 Progress

The Samba4 project started as an effort to rewrite Samba in a way that would allow such functionality - and it included a completely rewritten DCE-RPC (Remote Procedure Call) library. The development effort in creating this system has created the infrastructure[28] upon which an Active Directory compatible DC is entirely possible.

In developing Samba4, Tridgell [54] demonstrated the ability to join a Microsoft Windows XP Professional client to a Samba4 domain; however this process had not been independently reproduced. In Section 11.5 I proceed to reproduce this in my test environment.

1.3 Targeting Active Directory

In developing the new Samba4 infrastructure, the Samba Team decided to implement calls in the same way as Microsoft Windows 2003 Server. This decision is in line with a policy the Samba Team has consistently followed: to emulate the latest versions of Microsoft's products wherever possible. Because other software vendors test against the latest Microsoft releases, and because the latest Microsoft release is the only 'standard' we can reference, we must emulate that very closely.

This policy has been hampered in recent years by the significant changes made by Microsoft in developing Active Directory. The Samba and IBM Blue Directory research teams[2] determined that emulating parts of Windows 2000 would cause the client to assume Samba implemented other parts of the system. Since Active Directory is a fundamental part of the Windows 2000 (and later Windows 2003) architecture, this created an 'all or nothing' problem.

The Samba Team, lead by Tridgell [54], decided to remedy this problem in Samba4 by simply implementing the missing components.

1.4 Kerberos

The introduction of Kerberos with Windows 2000's Active Directory was the biggest change in Windows authentication since the introduction of Microsoft Windows NT. Seen as a fundamental shift away from pure password-based authentication schemes, the migration to Kerberos allowed Microsoft to adopt an industry-standard, extensible authentication system, with far more flexibility in implementation.

Though Kerberos, an industry-standard authentication system, was developed for Unix-like systems, Samba3 made very limited use of Kerberos.

[2] The work of this research team is described in Section 10.1.4.

Since Samba3 mirrored the un-kerberized NT4 almost all respects, users in a Samba3 domain were unable to enjoy all of the advantages Kerberos offers. Recent work has allowed Unix-like clients to use Kerberos in such domains[22], but Windows clients are still stuck with Microsoft's proprietary NTLM[16][3].

NT4 is now a legacy technology[63], despite the number of sites still running NT4 on both the client and the server. Therefore, the challenge is to emulate a much more recent version of Microsoft's offerings.

Because Kerberos is such key change in Active Directory, and because of the lack of documentation for NTLM, this thesis looks at the authentication problem in considerable detail.

1.5 More than had been done before

Support for the CIFS[4] protocol is clearly a characteristic element of Samba, but in Samba4 this is extended: Samba4 moves beyond CIFS to expose DCE RPC[5] over non-CIFS transports, such as directly on TCP/IP.

In previous versions of Windows, DCE RPCs were principally made over named pipes pipes, a concept existing in the file-system and shared between hosts by the CIFS file-sharing protocol. The change with Windows 2000 is that RPC became available directly on TCP/IP sockets.

In the same way that Samba must support the newer DCE-RPC over TCP, this thesis shows how Kerberos is added to the protocols that Samba4 supports, and to create patches to other projects to allow them to cooperate with Samba4 in implementing this.

[3] NTLM is an authentication scheme, described in Chapter 5

[4] CIFS is a network file-system, formerly known as SMB, that I describe in section 3.1

[5] DCE-RPC is a Remote Procedure Call system, i.e. a system that invokes functions over a network, to perform distributed computing. This is described in section 3.4

Part I

Background

Chapter 2

Introducing Active Directory

2.1 What is a directory server?

Directory servers are becoming a core component of most corporate networks: the central store for information about every computer and user as well as the basis for the network login system.

Directory servers are available from a number of vendors, and in their most basic form consist of key-value lookups on structured information. This information is often organised into a hierarchy.

The predominant directory access technology in use today is LDAP[17, 60], a descendant of the X.500 directory standard from the International Telecommunication Union (ITU), and a component of the full OSI networking stack.[24, 8].

2.2 What is Active Directory?

Active Directory is many things to many people:

2.2.1 Standards-based directory server

To those in institutions who care about such things, Active Directory is a standards-based (and in some ways standards-compliant) directory system, with access made possible by protocols such as LDAP and Kerberos. Both of these protocols provide views onto the directory, and the internal database appears to be a X.500 data model of some kind (but is not exposed to the outside world directly, only via LDAP).

By leveraging this standards base, Microsoft has been able to claim interoperability with other vendors. Unfortunately this is a one-way process, with Active Directory at the core, and other systems using Internet Standards to reference that data.

2.2.2 Proprietary directory server

While Microsoft has claimed compatibility with LDAP and Kerberos (being Internet standard technologies) to potential customers, a truthful analysis of how Microsoft's own clients use the AD system will show that this is for the benefit of others much more than Microsoft's own clients. The technology Microsoft uses for client-server communication is remote procedure calls (RPC). The use of RPCs in Windows networking has grown significantly since NT4.

These calls are not documented, even though the basic transport for them is DCE/RPC, for which documents and a reference implementation are available from The Open Group[41]. This documentation, however, does not address the format of individual calls that the client will make.

Much of the challenge of Samba4 is the continued task of 'network protocol analysis', the art of determining the operation of a proprietary network protocol by examining network operations[52].

2.2.3 Simple directory server

To small organisations, Active Directory is a system with which a Windows network can be administered from a single, simple, GUI interface. An Active Directory network may use LDAP and Kerberos, but these users simply do not need to know or care about this.

Chapter 3

Active Directory Protocols

Looking at Active Directory from the ground up, it is clear that a large number of network protocols are involved. Many of these are Internet standards, or widely understood, but others are proprietary.

3.1 CIFS

CIFS[27, 19, 49], perhaps the most important protocol in the Microsoft networking landscape, dominates the connections made between almost all clients and servers on a Windows network. Expanded, CIFS stands for the Common Internet File System, but the name only really tells us that it is a network file-system. Being a network file-system, files are shared over it, but unlike other network file-systems CIFS also presents printing and an Inter-Process Communication (IPC) interface. Accordingly CIFS carries much of the network activity in an Active Directory implementation.

3.1.1 CIFS, SMB and NetBIOS

There is much confusion and contradictory terminology in the area of CIFS networking. Originally SMB (Server Message Block), CIFS sits on top of the complete NetBIOS stack of services. Both of these are quite sufficient to fill a book,[1] but it is important to note that SMB and NetBIOS has historically run over IPX, DECNet and NetBEUI as well as TCP/IP.

3.1.2 CIFS as an IPC mechanism

CIFS exports the concept of 'named pipes' - a system for Inter-Process Communication (IPC) over the network, making CIFS a transport layer to RAP[2]

[1] Chris's Hertel's Implementing CIFSHertel [19] is a very good reference on the topic

[2] The Remote Administration Protocol (RAP) was implemented in LAN Manager, OS/2 and subsequently Windows NT but is now largely replaced by DCE-RPC.

and DCE-RPC in particular. This is unusual in networking, because both local and remote IPC are conducted in a very similar way, abstracting away many of the usual concerns. Because this CIFS transport layer is authenticated, and because CIFS (then SMB) ran over these multiple network layers, applications need not deal with either of these network transport issues.

Only with the release of Windows 2000 did Microsoft directly use TCP/IP as an IPC transport layer, in domain logon services.

3.2 LDAP

The Lightweight Directory Access Protocol (LDAP)[60, 59, 61, 20, 21, 58] has become the Internet standard for access to structured information, in particular information in the format of an X.500-like tree. Active Directory exports much of its information in the form of an LDAP tree, and a wide variety of tools are available to query and in some cases modify that information.

3.3 Connectionless LDAP

CLDAP (Connectionless LDAP) originally was an Internet standards-track proposal to allow LDAPv2 queries over UDP, a process that may be useful for service discovery. Microsoft implements CLDAP, but they do not follow the proposed standard, instead creating a version of CLDAP based on LDAPv3. As will be discussed in Section 8.3.1, Microsoft's CLDAP has become nothing more than an Internet Protocol transport for a proprietary blob of data. However, the implementation of Microsoft compatible CLDAP is important, because it forms a critical role in DC and domain location.

3.3.1 Sites

Sites are a concept closely tied to CLDAP and DNS - workstations will try to use servers found at their own site, rather than one further away (over potentially expensive WAN links). The CLDAP reply includes information on sites, and clients can use this, as well as packet timing information, to determine which DC to use.

3.4 DCE-RPC

DCE-RPC is a long-established standard for the operation of Remote Procedure Calls (RPC), and is published free of charge by the Open Group[41].

However, the complexity in DCE-RPC is not in the transport or basic operation (not that the difficultly in writing a DCE-RPC marshaling and control library should be underestimated), but in the proprietary security mechanisms and interface definitions.

3.4.1 Interface Definitions

Each function exposed over DCE-RPC has an associated interface definition, written in the Interface Definition Language (IDL), and if you were to make a particular interface public, all you would need to do is publish the IDL file. This is the same IDL file that you would compile yourself to create the initial framework and library on which you build your own client or server.

One of the big changes in Samba4 is PIDL, our own IDL compiler, detailed in Section 11.3.

3.4.2 DCE-RPC Security

NTLMSSP (described in Section 5.2) and Schannel (a similar security scheme between member workstations and domain controllers) are the two predominant security mechanisms applied to DCE RPC in a Microsoft environment, and both are considered proprietary by Microsoft. Fortunately there is a growing body of documentation on both, built up by independent researchers and the Samba Team. With the introduction of Active Directory, Kerberos also becomes available for DCE RPC. In all these cases the mechanism authenticates clients (by means of an authenticated 'bind') and can secure the traffic as it passes over the network.

3.4.3 Transports

DCE-RPC describes not only the format for the function calls, but also a number of network transports. The predominant network transport used in Windows networking is `ncacn_np` - the acronym is unimportant, but it means 'Named Pipes' and this file-system concept is carried over CIFS to remote servers. Other transports include:

`ncacn_ip_tcp` DCE-RPC over TCP/IP

`ncadg_ip_udp` DCE-RPC over UDP/IP

`ncacn_http` DCE-RPC over HTTP

3.5 DNS

Active Directory provides and uses an extensive amount of information in DNS, particularly based around the new SRV record type. Under the `_msdcs` sub-domain, Microsoft stores information about each of the domain controllers on the network, both by DC name and more importantly by the network service they provide.

Microsoft has devised their own secure update scheme for DNS, based around Kerberos. This allows machines to update their own entry in the DNS server.

3.6 SNTP

The Simple Network Time Protocol (SNTP) is an Internet Standard time protocol, and is used extensively in ADS domains to keep accurate time. Each domain controller runs a time server, and in an ideal network is synchronised with an upstream time server. In an interesting extension to SNTP, Microsoft signs the time responses with the schannel security scheme. This allows a secure bootstrapping to the correct time, needed for correct Kerberos operation.

Chapter 4

Authentication

4.1 Authentication problems

Authentication is about proving identity - proving that a particular person is who they claim to be, proving that a particular endpoint on a network is legitimately allowed to operate as a particular account. There are many issues surrounding the connection between a user and their network identity: enough certainly for a research paper in and of itself[9]. For the purpose of this discussion, a user will be considered to be the entity holding the password to an account. Likewise, a server is the entity that answers on a particular network address, and may hold its own server account details.

4.1.1 Who you are

The most common operation in the authentication area is proving a user's identity. In a common example, a logged on user would say, 'only I should be able to get at my mailbox'. This is an easy problem to solve in the abstract: you simply prove who you are, and proceed with other operations. Performing this operation in a manner that is secure to attack is difficult, however.

4.1.2 Who they are

One of the less-considered, but equally important matters in the authentication space is the authentication of peers. It is readily accepted that the user must prove their identity to a server, but with a few exceptions (mostly involving 'secure' websites), users do not expect a server to prove its identity to them.

Proof of server identity is a very important issue, as soon as any of the information given by that server needs to be trusted. Often called 'mutual authentication', a solution to this problem ensures that the server is 'trusted' not to provide malicious data, such as an invalid address book, and that it will behave properly with information, such as credit card numbers, given to it.

4.1.3 Data integrity

Once the identity of a server or client has been established, there must be some way to ensure that only that server or client can continue to communicate with the other. Otherwise, it may be possible for an impostor to take over a connection and impersonate one party to the other. This connection hijacking could allow unauthorized access to privileged documents, or server administration functionality.

The practice of data integrity, also known as 'signing', typically involves a cryptographic calculation where both sides can prove that: could only be performed by the other, and could only have been performed over the data received. If the signature (the output of that calculation) is invalid, the message must be considered compromised.

4.1.4 Data encryption

Whenever data flows over a communications network, it becomes open to attacks simply by observation. Should a user's password be sent in cleartext, an attacker could use it to later log in as that user. Likewise, if a confidential document were retrieved, an attacker in a position to watch the network traffic could also read that document.

The practice of data secrecy, also known as 'sealing', 'privacy' and 'confidentiality', involves the encryption of specific data portions, or the entire communications channel, such that only the other party can decrypt it.

4.2 Authorization problems

The problem of authentication and the problem of authorization are often lumped together, and this is certainly the case in Active Directory. However, we first need to consider them apart.

Once a user is authenticated (they are who they claim to be, or more simply they know the password to the account), a decision needs to be made as to what resources they may access: what files may they read/write, what hosts may they login to? This process is authorization, and even extends to such ideas as impersonation: a user may be authorized to impersonate another user.

The tempting thing to do when designing real-world systems it to embody a great deal of authorization information in the authentication process; information such as group membership may be returned as a product of the login process, and even evaluated to determine if an authentication attempt should succeed. This is frowned upon because it links the two processes very tightly: preventing a single authentication identity from having multiple authorization identities. That said, the two are linked (as they are in AD) for reasons of network efficiency.

4.3 Authentication Schemes

There are a large number of theoretical and practical ways to establish identity. Almost all rely on some degree of cryptography to secure the login process.

4.3.1 Plain-text Authentication

Certainly the easiest form of authentication to explain and implement, plaintext authentication turns up all over the modern networking world, from the typical use of HTTP Basic authentication[12] and HTTP form-based authentication to 'LDAP authentication', found inside many corporate networks. Typically, these are not protected by a transport layer security (a security mechanism that protects the entire stream, not just the passwords), and are otherwise clear on the network. If the remote server is not somehow authenticated, it could maliciously indicate successful authentication or steal the passwords.

4.3.2 Challenge-response Authentication

Challenge-response authentication is typically a shared-secret scheme, where both parties to the authentication exchange have a copy of the password, or a fixed derivative thereof. As shown in Figure 4.1, the server generates a random 'challenge' to the client, and asks the client to perform a fixed operation with inputs consisting of the 'challenge', the user's password, and possibly some random data of the client's choosing.

The result of this operation should not in any way disclose the user's password, and should be repeatable on the server. Figure 4.2 shows how, when the server repeats the operation using its copy of the password, it compares the output with the value supplied by the client. If the values match, the client must know the user's password.

NTLM (described in chapter 5)and Kerberos (described in chapter 6) are both challenge-response authentication schemes, and are both used extensively in Active Directory. However, Kerberos has important other properties that make distributed operation far more secure than NTLM, an area discussed in section 5.3.

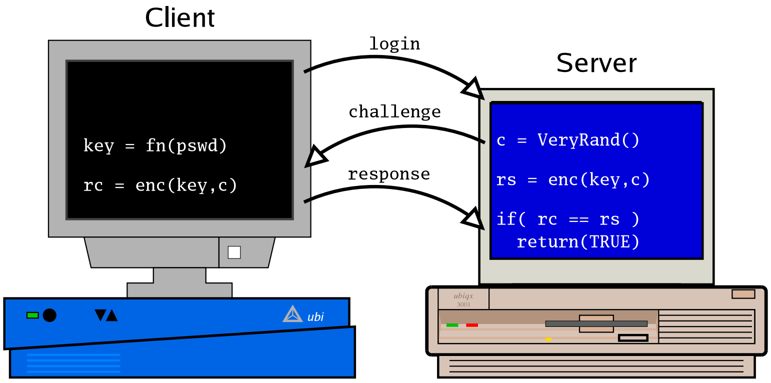


Figure 4.1: Challenge/Response

The server generates a random challenge, which it sends to the client. Both systems encrypt the challenge using the secret encryption key. The client sends its result (rc) to the server. If the client's result matches the server's result (rs), then the two nodes have matching keys. (Image and text (c) Chris Hertel[19], http://www.ubiqx.org/cifs/figures/smb-11.html)

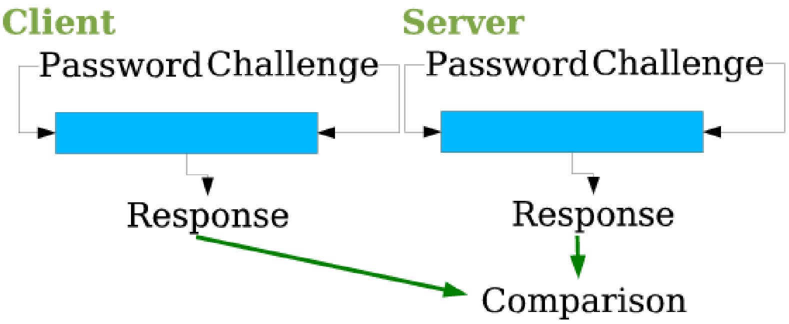


Figure 4.2: The server must compare it's calculated result with the one supplied by the client

4.3.3 Trusted Third Party Authentication

Many distributed authentication systems allow logins to occur on numerous hosts, but only a few hosts (possibly one) actually confirm or deny an authentication request. These are trusted third party systems; all hosts trust those with the passwords (the third party in the authentication exchange) to correctly return authentications success or failure.

For an authentication system to be secure, it must be possible to trust this third party, preferably by some cryptographic proof. Often this is by yet another shared-secret authentication scheme.

4.3.4 Single Sign On

Often abbreviated as simply SSO, the concept of Single Sign On is a matter of usability; users wish to establish their identity once, and not have to think about it after that. This allows for more complex authentication procedures as the user only has to tolerate them once per session. SSO has become the expectation in modern network environments.

Chapter 5

NTLM

5.1 NTLM Challenge Response

NTLM is a challenge-response authentication scheme, designed to prevent a direct compromise of the user's password as it passes from client to server. In theory, the value that the client gives the server can only be generated with knowledge of the password, but does not reveal the password itself.

More importantly, this authentication scheme has a direct heritage back to the early days of the CIFS (then SMB) networking suite, with much of the weakness in it as an authentication scheme linked to backward compatibility. It is the ubiquitous authentication scheme in Windows networking, and even with Windows 2000 is the transparent fall-back option if and when Kerberos fails to operate in a particular environment.

5.1.1 Shared Secrets and Hashing

NTLM is a shared secret authentication scheme. The shared secrets are hashed (passed though a one-way-function), so as not to be the original plain-text password, but still password equivalent for the purpose of challenge-response authentication. As is implied by the name, the shared secrets are stored on disk on the Server, and derived from the password typed in by the user.

There are two password hashes in NTLM, known as the NT hash (introduced with Windows NT) and the LM or LANMAN hash (compatible with LAN Manager) are 16 byte quantities[33]

Calculating the LM Hash

The LM hash is constructed from the uppercase, ASCII version of the user's password.

The password is truncated/NULL padded to 14 ASCII characters[1], and split into two 7 byte halves. Each half is used as a key to encrypt the constant string "KGS!%@$%". To the extent that DES is a strong encryption functionSchneier [47], neither knowledge of this 'plaintext', nor the encrypted output allows one to find the key. Therefore this is a one-way hash function.

The function is (in pseudocode, C-style 0 based arrays):

```
part1 = DES(upper(password[0-6]), "KGS!%@$%")
part2 = DES(upper(password[7-13]), "KGS!%@$%")
LMhash = concat(part1, part2)
```

The fact that the password is constructed in two halves (as well as properties of the LM challenge response function below) creates a cryptographic weakness in the security of LM challenge-response authentication. This is discussed in Section 2.8.3.5 of Implementing CIFS[19], but in short, the password's length (less than, equal to or greater than 8 characters) is exposed in the final output.

Calculating the NT Hash

The NT hash is much stronger, and quite simply created:

```
NThash = MD4(unicode(password))
```

Because the NT hash is not composed in parts, and uses the secure hash algorithm MD4[42], it is 128 bits in strength. It is also Unicode, allowing for a wider variety of passwords, particularly in languages other then English. Most importantly however, it is mixed case, lacking the forced uppercase step of the LM hash.

5.1.2 LM Challenge Response

Sharing the secrets is only one half of a shared-secret based authentication scheme - the more important half is securely proving that you know that secret. This is where the LM challenge-response scheme comes in, with another round of DES applied to either the NT or LM hash. This process is discussed in detail, including an example implementation, in Section 2.8.3.4 of Hertel [19], but is summarised here.

The server generates an 8 bytes cryptographic challenge (the LM challenge), consisting of random data.

The LM response takes successive 7 byte parts of the NT or LM hash, and uses them to encrypt that LM challenge:

[1] Implementations should not store/disable any LM password created from more than 14 ASCII characters, as use of this password would compromise the additional length.

```
    resp1 = DES(hash[0-6], chall);
    resp2 = DES(hash[7-13], chall);
    resp3 = DES(concat(hash[14-15], zeros[5]), chall);
    resp = concat(resp1, resp2, resp3);
```

The 24 byte response (8 bytes from each DES operation) is then sent over the network, proving to the server that the client knows the NT or LM hash, and presumably therefore the password. This is known as the LM response when based on the LM hash, and the NTLM response when based on the NT hash.

The LM challenge-response function sufferers from a number of flaws, but the most easily understood is that it is 56 bit DES at best. Other weaknesses (the 7 byte split in the underlying LM response) make it possible to reverse the calculations in a period of hours on modern hardware.[39]

5.1.3 NTLMv2

Due to the problems with the security of the NTLM challenge-response scheme, a new scheme was devised by Microsoft, and became known as NLTMv2[35]. Unfortunately a number of other improvements are also labeled NTLMv2[2], but we will start by describing the new NTLMv2 challenge-response:

NTLMv2 uses the same NT hash, but instead of the LM challenge-response formula, a new system based on HMAC-MD5Krawczyk et al. [26] has been built. Because the 56-bit (cypher strength) DES step has been replaced by a 128-bit HMAC-MD5 function, logins can be considered to be protected by 128 bit security.

5.1.4 Session Keys

As a byproduct of NTLM authentication, a password-derived ‘session key’ is produced for use in verifying or encrypting data carried between the client and server. The algorithm used varies depending on the method of authentication and unfortunately can be very weak - often a fixed derivative of the user’s password! This key is known as the ‘user session key’, and is used in a number of places within CIFS directly, as well as by the NTLMSSP suite.

LM session key construction

The LM session key is constructed from the first 8 bytes of the LM hash, padded to 16 bytes with zeros. Given what we understand about the LM

[2] The improvements labeled as NTLMv2 include the NTLM2 session response (described in Section 5.2.2) and NTLMv2 Session Security, which changes the NTLMSSP signing and sealing algorithm, (described in section 5.2.3).

hash, it is equivalent to the password for passwords of 7 characters or less!

```
LM_key = concat(head(LM_hash, 8), zeros[8]);
```

NT session key construction

The NT session key is also fixed derivative of the password:

```
NT_key = md4(NT_hash);
```

Which is also

```
NT_key = md4(md4(unicode(password)));
```

5.2 NTLMSSP

NTLMSSP is a collection of protocols, which together fulfil the Microsoft Security Support Provider Interface (SSPI[32]). As such, the NTLM challenge-response steps have been wrapped into a framework such that a calling application need only know how to pass these messages, not to understand them. At each end of the connection, these messages are passed down to the security libraries for processing.

5.2.1 NTLMSSP Packets

Within those messages is a particular packet format, which is known as NTLMSSP, partly because the ASCII string `"NTLMSSP"` prepends every protocol exchange. Three different packets pass back and forth between client and server:

Negotiate The initial packet, sent from the client to the server, suggesting options (including choice of Unicode or ASCII for future communication) and requesting an authentication

Challenge The return packet, containing the LM challenge, and the server's options (influenced by the client). It may also include data on the target system's name and domain.

Authenticate The final packet, containing the user-name, domain and challenge-response (also known as the encrypted passwords), in whatever format may have been negotiated.

The format of these packets, and the meaning of most of the options carried in them is now reasonably well understood, and partially documented in an official sense by Open Group [40]. Unofficial documentation includes the comprehensive reference by Glass [16] and the older reference by Tschalar [55].

5.2.2 NTLMSSP Options

Within the NTLMSSP context, a different set of session keys, cyphers and authentication inputs are used - depending on the negotiated options. The fact that these are negotiated is problematic, but the implementation may define minimum required options. What follows is a discussion of some of the options, but a more complete treatment is given by Glass [16].

LM Session Key

The LM session key is created as specified by Open Group [40]: it is based on the NTLM 'LM Key', and includes part of the LM response (and therefore the server-generated random challenge) in a DES based hash, making it unique for each session. It is negotiated by the specification of the `NTLMSSP_NEGOTIATE_LM_KEY` in the negotiated options.

This key is then 'weakened' to various strengths, to fix export requirements. The irony is that the 128 bit negotiated key is far from this in real strength, due to there being at most 56 bits of key input!

NT Session Key (v1)

When the LM_KEY option is not negotiated, and no other options are specified, the session key is the NT Key from the NTLM authentication exchange. This key is 128 bits in strength, but fixed until the user's password changes. Unfortunately, despite being 128 bits in strength, the session key's value can be obtained by breaking the 56-bit DES in the LM response algorithm, so it never provides 128 bits of cryptographic protection.

NTLM2 Session Response

Another modification to the NTLMSSP login scheme, this option prevents a server-initiated dictionary attack, by providing input from the client and server in calculating the challenge input to the challenge-response function. This option also changes the session key negotiation to include mutually agreed random data into the key. This ensures that the session key again changes between sessions.

Key Exchange

In another modification to the session key negotiation, the specification of the 'key exchange' flag allows the client to specify a new session key, to be encrypted with what otherwise would be the session key. Presumably, the client would choose a random sequence of bytes, unrelated to the password, but as will be noted in Section A.3, the ability for the client the propose a known session key is an unexpected weakness in the NTLMSSP scheme, particularly given the steps taken when the NTLM2 Session Response is selected.

5.2.3 NTLMSSP Signing and Sealing

NTLMSSP provides generic functions to sign and seal quantities of data, and this is used in DCE-RPC. Using the session key negotiated between the client and server, a cipher state is established though which the data is encrypted, or the signature is generated/verified.

The core of the sign/seal algorithms is alleged to be RC4[44, 29, 47]: an algorithm used repeatedly (and unfortunately often poorly) in many parts of Microsoft's network server implementation. When negotiating the newer 'NTLM2 Session Security', multiple RC4 sboxs are generated, one per direction, each using unique subkeys, and checksums are calculated with the strong HMAC-MD5[26] checksum. However, when this is not negotiated the checksum is a only a CRC32 function[64]; weaknesses in this function have been used in the past to break the security of SSH connections[15].

5.3 Distributed NTLM

NTLM as so far described has no inbuilt mechanism for distribution over a network. In short, NTLM assumes that the server being contacted is also the server holding the passwords. In order to implement a distributed network architecture, compromises, which are invisible to the client, must be made at the server. Typically these are to somehow contact the Domain Controller (DC) to confirm or deny an incoming user's identity, a process shown in 5.1.

5.3.1 Pass-though Authentication

The first, and easiest compromise the server can make is simply to defer the decision to another server. In pass-though authentication, optimised by Samba's `security=server`[56] configuration option, the challenge is generated by a remote server, and the response forwarded to it likewise. The server in the middle, to which the client wants access, simply assumes that the remote server will correctly signal authentication success or failure. Such a server does not gain any additional knowledge, such as session keys, nor does it have cryptographic proof that the remote server is not in some way being spoofed.

This mode of operation is particularly fragile, because once a challenge is retrieved from the remote server, the connection *must* remain open - a transparent re-connection cannot be made later.

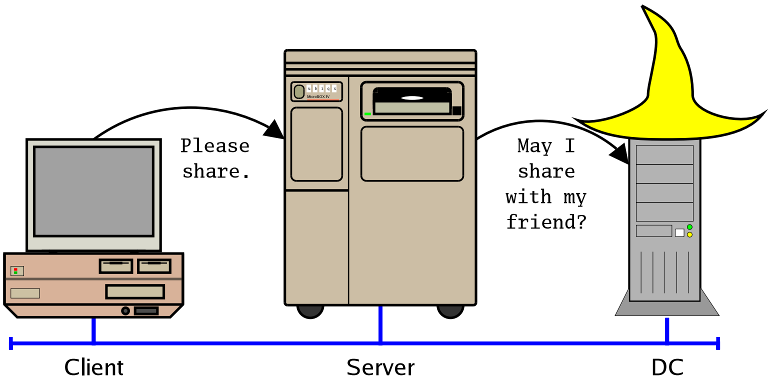


Figure 5.1: Trusted Third Party Authentication (NTLM).

> The Domain Controller (DC) wears a special hat. It keeps track of the common authentication database that is shared by the SMB servers in the Domain. The SMB servers query the DC when a client requests access to SMB services. (Image and text (c) Chris Hertel[19], `http://www.ubiqx.org/cifs/figures/smb-15.html`)

5.3.2 NETLOGON

The NETLOGON authentication process was introduced by Microsoft Windows NT to handle this problem in a more secure manner. Fundamentally, the challenge-response step to the remote server (the DC) is removed, and instead the DC is presented with both the challenge and the response. The server in the middle (the member server) gains in return a set of details about the user: their name, groups and the cryptographic session keys described above.

However, this process presents a security exposure - why couldn't an attacker ask the DC the same question, perhaps with a challenge and response found by sniffing the network? And why couldn't an attacker modify the response from the DC to include the 'Administrators' group? This is prevented (on later clients and servers[3]) by the cryptographic signing and sealing of the connection, using a shared secret between the member server and the DC.[4]

It is for the purpose of setting up this secret that a machine 'joins' the domain in the first place.

[3] Windows NT4 with Service Pack 4 and above, Windows 2000 and above and Samba 3.0 and above.

[4] This is a system known as schannel or Secure Channel.

While modern clients will negotiate, and even require,[5] a level of security in their communication with the DC, another issue remains very real: the circle of friends that the DC trusts to perform this operation is very large, extending to each and every workstation that is a member of the domain, not just the file and print servers. Any single one of these may perform this logon operation, and obtain information such as the user's long term session key, or even the first 8 bytes of their LANMAN password!

[5] Windows XP will not communicate with a Domain Controller that does not support the schannel scheme for signing and sealing it's connection to the DC.

Chapter 6

Kerberos

6.1 Kerberos Basics

Kerberos[47] is described as a 'trusted third party' authentication system, based around shared secrets and symmetric cryptography. The hub of the Kerberos authentication system is the Key Distribution Center (KDC), which contains a copy of each user's passwords (in very much the same way a DC or standalone server does for NTLM). Kerberos is an Internet Standard[25], with MIT providing the reference implementations, used in their Project Athena. In an interesting diversion from the standards documents, four-part dialogue, describing the system is available[6].

Aside from a far superior cryptographic base, the key difference between NTLM and Kerberos authentication is that the user is given their ticket, with cryptographic proof of their own identity. The server can verify that proof with local cryptography, avoiding the need for the server to contact the KDC in real time. Kerberos also includes mutual authentication, signing and sealing of data, and is extensible with new encryption types, something that Microsoft did allow in an upgrade from Windows NT:

6.2 An upgrade compatible encryption type

One of the biggest deterrents to the deployment of Kerberos at any site is the frightening concept that to do it, all of the user's passwords must be securely collected, in clear-text, for hashing into appropriate Kerberos keys. Microsoft knew that it could not expect this kind of effort in upgrading Windows servers from NT4 to Active Directory, so they solved it a different way.

Instead of obtaining the plain-text password, Microsoft used the NT hash[1] as the shared Kerberos key, and defined a set of cryptographic op-

[1] Previously calculated and stored for use with the NTLM password scheme

erations around this base. Published as an Internet-Draft Brezak and Swift [4], this encryption type has been added to other Kerberos implementations for the purpose of interoperability.

6.3 The PAC

Kerberos follows a model that describes a strict separation between authentication and authorization. This is very nice in theory, but presents some practical difficulties in mapping a Kerberos identity to a user, and their access rights. In particular, it is the problem of performing this mapping in a network-efficient manner that caused the PAC (Privilege Attribute Certificate) and associated infrastructure to be devised.

The PAC is a cryptographically signed blob of data including information on a user's groups, their home directory location, and similar details. This is nothing particularly special, but Microsoft created a storm by releasing the specification for this data format under a 'click-though' licence, including a Non-Disclosure Agreement (NDA)[62].

Subsequently, portions of this data format were released under a much more liberal licence[5].

Chapter 7

Security Negotiation

In what is in many ways an oxymoron, security protocols must be negotiated between any two peers, to ensure that each site is talking a dialect (or indeed and entire language) that the other understands. Invariably, these negotiation protocols are described as simple,[1] and provide each side with choice over the desired mechanisms. The idea of security negotiation is also associated with security fall-back; if one authentication scheme is unavailable (even without a negotiation step), the peers may simply attempt an older scheme. This too has security implications, if an attacker can influence the decision to fall-back.

Once a mechanism is chosen, it may also perform its own negotiation steps, as NTLMSSP does with the negotiated flags.

7.1 SASL

As far as security negotiation mechanisms go, SASL is surprisingly sane. SASL succeeds because it is very easy to implement in almost any protocol: it places no requirement on the use of ASN.1, and is often trivially placed into text-based Internet protocols such as IMAP and SMTP. Security mechanisms are advertised and selected by simple exchange of text strings. By convention (and specified in standards for each protocol it is actually placed in), the subsequent exchange of security packets are again simple text strings, by simple use of base64 encoding.

Away from protocol implementation details, the names of security mechanisms are associated with particular levels of security. Clients and servers may place requirements on each other by which mechanisms they support, but this does not have any network-visible artifacts.

[1] A point made very good fun of by Allison [1].

7.2 GSSAPI

Designed as a simple API for access to Kerberos and other security mechanisms, GSSAPI is both a C programming API, and a network protocol. Both are reasonably complex, a point exampled well by the way GSSAPI uses an Object IDentifier (OID) to prefix its network messages. OIDs are globally unique streams of numbers, delegated out of a hierarchical namespace, and formatted (as is the case for all of GSSAPI) in ASN.1. (SASL uses simple text strings for the same purpose, with much clearer effect).

GSSAPI, like SASL, exchanges datagrams until both sites are happy with the security result.

7.2.1 SPNEGO

Born out of the need to mutually negotiate a security mechanism, where both parties may not already know what the other supports, the Simple and Protected NEGOtiation protocol (SPNEGO) is a selection mechanism for GSSAPI. It appears in a GSSAPI wrapping, because on the network it is a GSSAPI mechanism. The server or client (depending on who 'speaks' first) suggests a list of mechanisms, and the peer responds. If both parties agree on a mechanism, then that mechanism's exchange can commence. By anticipating that choice, the exchange for one of these mechanisms can be started in the negotiation phase, providing less round trips.

SPNEGO is interesting because it is used by Microsoft to select between Kerberos and NTLMSSP, and is the basis for their 'Negotiate' SSPI security mechanism. Microsoft also broke their implementation of SPNEGO, by removing the integrity protection. (This allows an attacker to modify the list of protocols being exchanged, to force a selection of a less secure protocol).

Chapter 8

The Join process

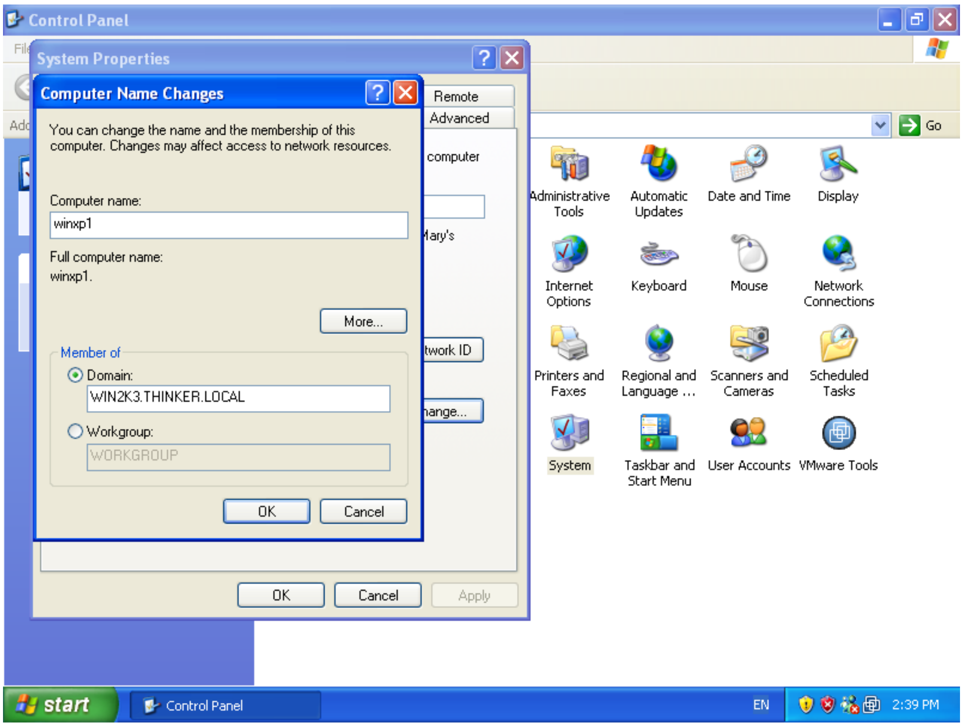
8.1 Purpose

The purpose of the 'domain join' is to securely setup a password (shared secret) between the workstation (or member server) and the domain controllers, so it may use the NETLOGON mechanism described in 5.3.2, and the Kerberos system described in 6. This is done by a privileged user, who has the right to specify that a new machine account be added to the domain. At the conclusion of this process, both the workstation and the domain controllers know the password, and can use this value to prove to each other that they are indeed authentic.

8.2 Process and Tools

In this example, a WinXP Professional, Service Pack 2, virtual workstation is joined to Active Directory, hosted on Windows 2003 Advanced Server. The workstation is a virtual machine running under VMWare [57], and is known in this process as `WINXP1`. The domain it is joining is `WIN2K3.THINKER.LOCAL` with `WIN2003.WIN2K3.THINKER.LOCAL` being the Windows 2003 Domain Controller (DC). The `Administrator` user is used to join the machine to the domain, which also has a short name of `WIN2K3THINK`. This was monitored[1] and analysed with the assistance of Ethereal [11], which produced the packet capture diagrams.

[1] Because both the WinXP and Windows 2003 machines are running under VMware (which itself is run on Linux), the whole process can be monitored by listening on the virtual Ethernet hub that VMware provides.



Computer Name Changes

You can change the name and the membership of this computer. Changes may affect access to network resources.

Computer name: winxp1

Full computer name: winxp1.

Member of

● Domain: WIN2K3.THINKER.LOCAL

○ Workgroup: WORKGROUP

Figure 8.1: The Join Dialog

8.3 Initial Join

The join process is initiated by selecting ‘Change’ from the ‘Identity’ tab of the ‘System’ control panel, on the WinXP client. This presents a very simple interface, shown in Figure 8.1.

8.3.1 DC Location

The first part of the domain join process is to locate a Domain Controller (DC) to join. This involves both DNS and CLDAP, if available.

DNS

Microsoft has defined a number of expected DNS names, under the `_msdcs` sub-domain, containing SRV records, for service location. The first assumption for a domain join is that the domain exists as a DNS name, with `_ldap._tcp._msdcs.` prepended.

```
No.     Time        Source                Destination           Proto
col Info
    182 357.875327  192.168.3.129         192.168.3.2           DNS
    Standard query SRV _ldap._tcp.dc._msdcs.WIN2K3.THINKER.LOCAL

Frame 182 (101 bytes on wire, 101 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:50:56:f4:6b:f8
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.2 (192.168.3.2)
User Datagram Protocol, Src Port: 1036 (1036), Dst Port: domain (53)
Domain Name System (query)
  Transaction ID: 0x13d5
  Flags: 0x0100 (Standard query)
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    _ldap._tcp.dc._msdcs.WIN2K3.THINKER.LOCAL: type SRV, class inet
      Name: _ldap._tcp.dc._msdcs.WIN2K3.THINKER.LOCAL
      Type: Service location
      Class: inet
```

Figure 8.2: Initial DNS Lookup

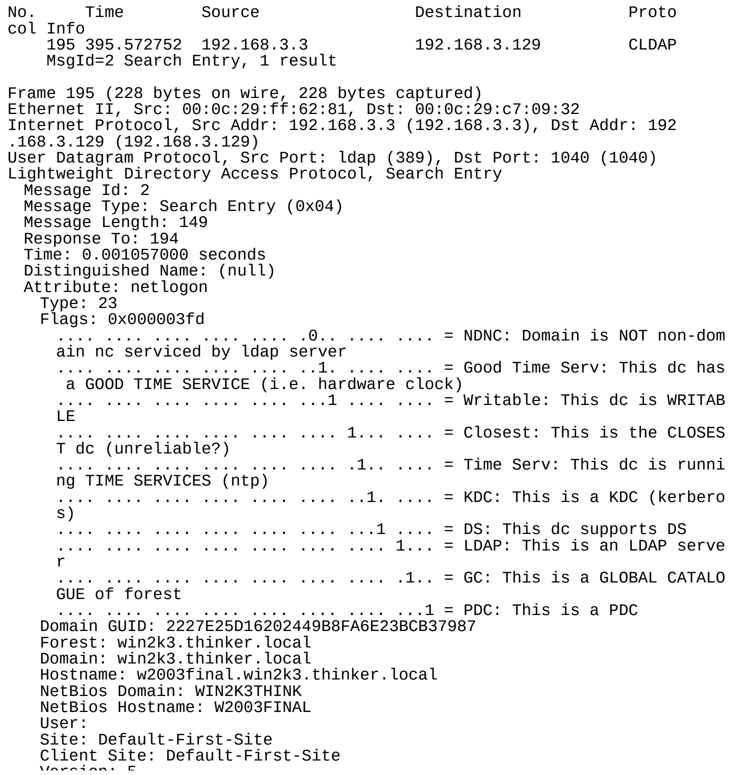
```
No.     Time        Source                Destination           Proto
col Info
    188 358.033510  192.168.3.129         192.168.3.3           CLDAP
    MsgId=1 Search Request, Base DN=(null)

Frame 188 (174 bytes on wire, 174 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:0c:29:ff:62:81
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.3 (192.168.3.3)
User Datagram Protocol, Src Port: 1038 (1038), Dst Port: ldap (389)
Lightweight Directory Access Protocol, Search Request
  Message Id: 1
  Message Type: Search Request (0x03)
  Message Length: 117
  Response In: 189
  Base DN: (null)
  Scope: Base (0x00)
  Dereference: Never (0x00)
  Size Limit: 0
  Time Limit: 0
  Attributes Only: False
  Filter: (&(DnsDomain=WIN2K3.THINKER.LOCAL)(Host=WINXP1)(NtVer=\006)
  Attribute: Netlogon
```

Figure 8.3: Initial CLDAP Lookup

CLDAP

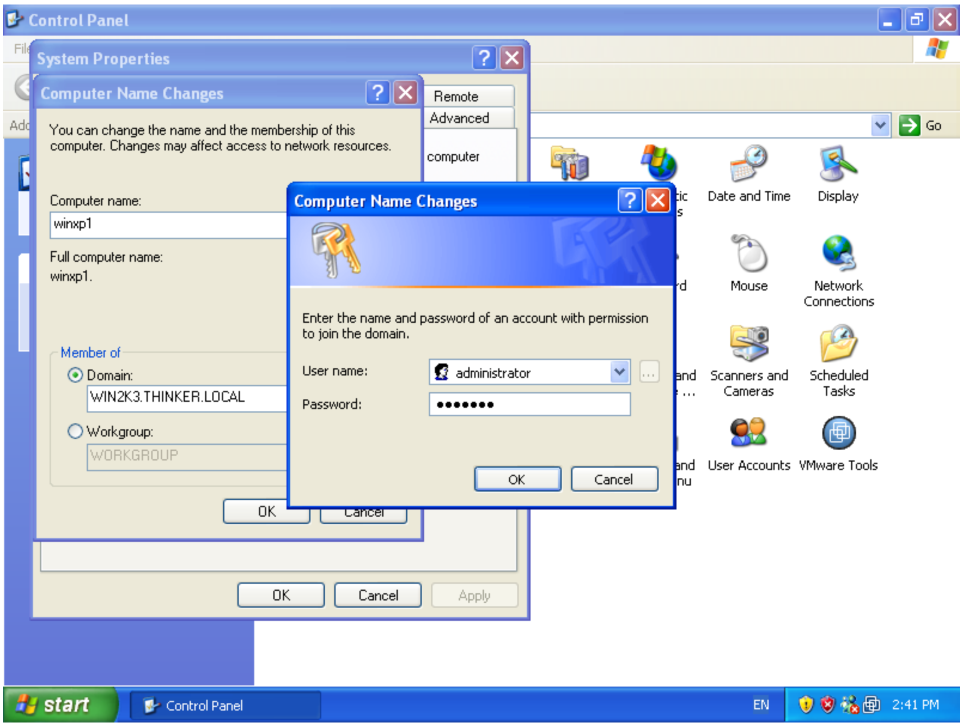
Connectionless LDAP is used as an ‘Internet Standard’ transport for Microsoft’s proprietary domain controller location protocols. While CLDAP would be an ideal mechanism for this purpose, it has been mangled into not much more then a wrapper for a proprietary network structure in the form of the `netlogon` attribute. Information about the machine attempting to locate the domain is included as an LDAP filter, and is sent to hosts appearing in the DNS reply. Part of the purpose of this process is to easily eliminate bad DNS data from the following processes: only those servers that send a reply to the UDP based CLDAP need be considered for further connection attempts.

The Netlogon CLDAP reply includes a large amount of detail about the server, its basic operating details and in particular the site at which it is located. Windows clients in an Active Directory domain can use the site information to restrict operations to domain controllers located at the same physical site as the client, avoiding stress on often comparatively slow



```
No.     Time        Source                Destination           Proto
col Info
    195 395.572752  192.168.3.3           192.168.3.129         CLDAP
    MsgId=2 Search Entry, 1 result

Frame 195 (228 bytes on wire, 228 bytes captured)
Ethernet II, Src: 00:0c:29:ff:62:81, Dst: 00:0c:29:c7:09:32
Internet Protocol, Src Addr: 192.168.3.3 (192.168.3.3), Dst Addr: 192
.168.3.129 (192.168.3.129)
User Datagram Protocol, Src Port: ldap (389), Dst Port: 1040 (1040)
Lightweight Directory Access Protocol, Search Entry
  Message Id: 2
  Message Type: Search Entry (0x04)
  Message Length: 149
  Response To: 194
  Time: 0.001057000 seconds
  Distinguished Name: (null)
  Attribute: netlogon
    Type: 23
    Flags: 0x000003fd
      .... .... .... .... .... .0.. .... .... = NDNC: Domain is NOT non-dom
      ain nc serviced by ldap server
      .... .... .... .... .... ..1. .... .... = Good Time Serv: This dc has
       a GOOD TIME SERVICE (i.e. hardware clock)
      .... .... .... .... .... ...1 .... .... = Writable: This dc is WRITAB
      LE
      .... .... .... .... .... .... 1... .... = Closest: This is the CLOSES
      T dc (unreliable?)
      .... .... .... .... .... .... .1.. .... = Time Serv: This dc is runni
      ng TIME SERVICES (ntp)
      .... .... .... .... .... .... ..1. .... = KDC: This is a KDC (kerbero
      s)
      .... .... .... .... .... .... ...1 .... = DS: This dc supports DS
      .... .... .... .... .... .... .... 1... = LDAP: This is an LDAP serve
      r
      .... .... .... .... .... .... .... .1.. = GC: This is a GLOBAL CATALO
      GUE of forest
      .... .... .... .... .... .... .... ...1 = PDC: This is a PDC
    Domain GUID: 2227E25D16202449B8FA6E23BCB37987
    Forest: win2k3.thinker.local
    Domain: win2k3.thinker.local
    Hostname: w2003final.win2k3.thinker.local
    NetBios Domain: WIN2K3THINK
    NetBios Hostname: W2003FINAL
    User:
    Site: Default-First-Site
    Client Site: Default-First-Site
    Version: 5
```

Figure 8.4: Initial CLDAP Reply



Computer Name Changes

You can change the name and the membership of this computer. Changes may affect access to network resources.

Computer name: winxp1

Full computer name: winxp1.

Member of

● Domain: WIN2K3.THINKER.LOCAL

○ Workgroup: WORKGROUP

Computer Name Changes

Enter the name and password of an account with permission to join the domain.

User name: administrator

Password: •••••••

Figure 8.5: Prompting for a password

WAN links.

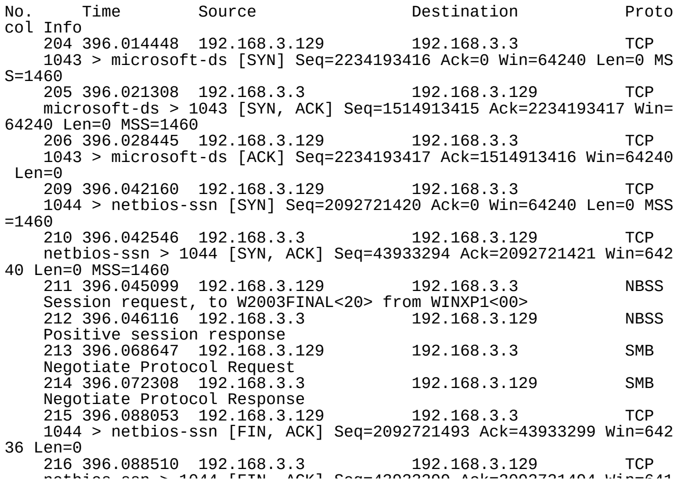
NetBIOS Lookups and NetBIOS GetDC

If a DNS name is not found, an NT4 style NetBIOS lookup can be performed, looking for the `WIN2K3THINK#1c` NetBIOS name, and using the result in the same way as the DNS result is used above. Likewise, a GetDC request (this time encapsulated in the NetBIOS mail-slot protocol), will return the same proprietary blob of information regarding the domain.

8.3.2 Asking for a password

Now that the Domain and Domain Controllers have been located[2], the user is prompted for a password, shown in Figure 8.5. This is the last thing a user sees before the process is completed, only a few seconds later. From here on, we can only observe progress by watching the network traffic.

[2] In particular, this step ensures that the domain exists, and the user has not made a typographical error



```
No.     Time        Source                Destination           Proto
col Info
    204 396.014448  192.168.3.129         192.168.3.3           TCP
    1043 > microsoft-ds [SYN] Seq=2234193416 Ack=0 Win=64240 Len=0 MS
S=1460
    205 396.021308  192.168.3.3           192.168.3.129         TCP
    microsoft-ds > 1043 [SYN, ACK] Seq=1514913415 Ack=2234193417 Win=
64240 Len=0 MSS=1460
    206 396.028445  192.168.3.129         192.168.3.3           TCP
    1043 > microsoft-ds [ACK] Seq=2234193417 Ack=1514913416 Win=64240
 Len=0
    209 396.042160  192.168.3.129         192.168.3.3           TCP
    1044 > netbios-ssn [SYN] Seq=2092721420 Ack=0 Win=64240 Len=0 MSS
=1460
    210 396.042546  192.168.3.3           192.168.3.129         TCP
    netbios-ssn > 1044 [SYN, ACK] Seq=43933294 Ack=2092721421 Win=642
40 Len=0 MSS=1460
    211 396.045099  192.168.3.129         192.168.3.3           NBSS
    Session request, to W2003FINAL<20> from WINXP1<00>
    212 396.046116  192.168.3.3           192.168.3.129         NBSS
    Positive session response
    213 396.068647  192.168.3.129         192.168.3.3           SMB
    Negotiate Protocol Request
    214 396.072308  192.168.3.3           192.168.3.129         SMB
    Negotiate Protocol Response
    215 396.088053  192.168.3.129         192.168.3.3           TCP
    1044 > netbios-ssn [FIN, ACK] Seq=2092721493 Ack=43933299 Win=642
36 Len=0
    216 396.088510  192.168.3.3           192.168.3.129         TCP
    netbios-ssn > 1044 [FIN, ACK] Seq=43933299 Ack=2092721494 Win=641
```

Figure 8.6: Initial CIFS connection

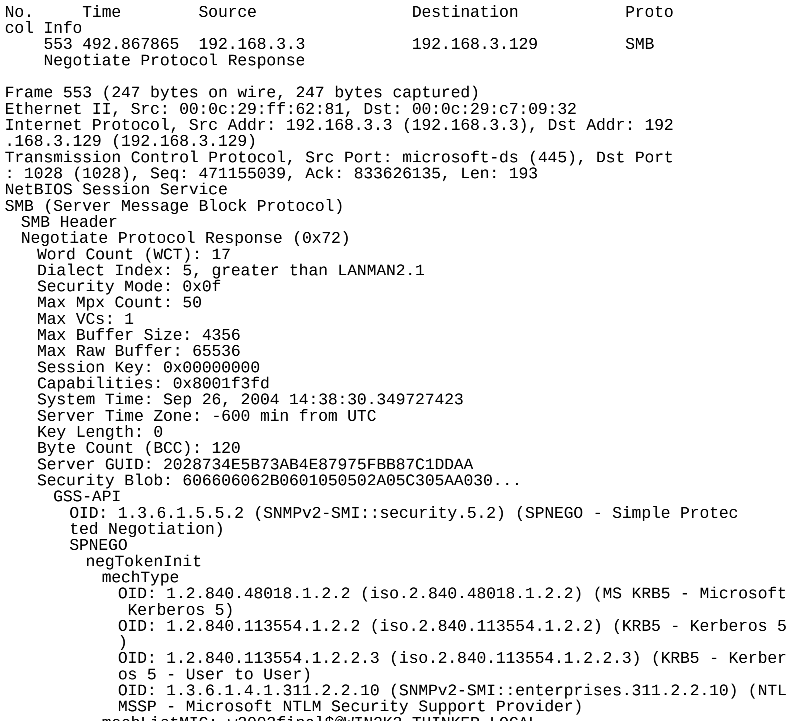
8.3.3 CIFS Connection

Two connections!

The address found in DNS is used to make a connection to the two registered CIFS ports on the server, and is shown in Figure 8.6. If both ports successfully connect, then the connection to port 139 is dropped, with the connection continuing on port 445. This strange behaviour is justified by the removal of NetBIOS (which is connected with port 139) from the CIFS protocol stack - in a mode that Chris Hertel[19] likes to refer to as 'naked transport'. Clearly, the interest here is not in network efficiency[3].

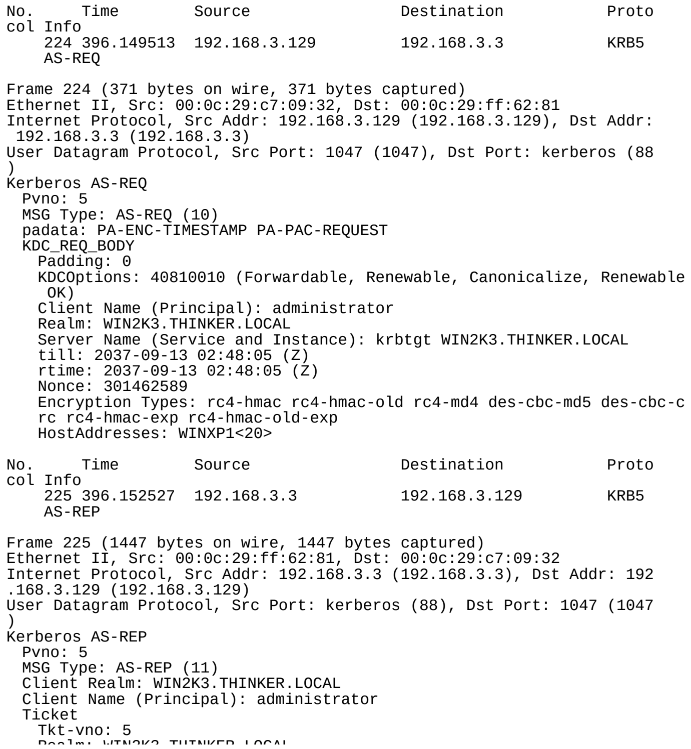
Negotiated Security

The expanded packet in Figure 8.7 shows the initial SPNEGO Surati and Muckin [50] token, as part of the Negotiate Protocol reply, indicating what security mechanisms this server is able to accept. The presence of Kerberos (in two forms, due to an unfortunate bug in early Windows 2000 versions), allows the login to progress using Kerberos authentication. Likewise, a client unable to perform Kerberos knows from this reply that NTLMSSP, the more traditional login scheme on Windows networks, is also acceptable.



```
No.     Time        Source                Destination           Proto
col Info
    553 492.867865  192.168.3.3           192.168.3.129         SMB
    Negotiate Protocol Response

Frame 553 (247 bytes on wire, 247 bytes captured)
Ethernet II, Src: 00:0c:29:ff:62:81, Dst: 00:0c:29:c7:09:32
Internet Protocol, Src Addr: 192.168.3.3 (192.168.3.3), Dst Addr: 192
.168.3.129 (192.168.3.129)
Transmission Control Protocol, Src Port: microsoft-ds (445), Dst Port
: 1028 (1028), Seq: 471155039, Ack: 833626135, Len: 193
NetBIOS Session Service
SMB (Server Message Block Protocol)
  SMB Header
  Negotiate Protocol Response (0x72)
    Word Count (WCT): 17
    Dialect Index: 5, greater than LANMAN2.1
    Security Mode: 0x0f
    Max Mpx Count: 50
    Max VCs: 1
    Max Buffer Size: 4356
    Max Raw Buffer: 65536
    Session Key: 0x00000000
    Capabilities: 0x8001f3fd
    System Time: Sep 26, 2004 14:38:30.349727423
    Server Time Zone: -600 min from UTC
    Key Length: 0
    Byte Count (BCC): 120
    Server GUID: 2028734E5B73AB4E87975FBB87C1DDAA
    Security Blob: 606606062B0601050502A05C305AA030...
      GSS-API
        OID: 1.3.6.1.5.5.2 (SNMPv2-SMI::security.5.2) (SPNEGO - Simple Protec
        ted Negotiation)
        SPNEGO
          negTokenInit
            mechType
              OID: 1.2.840.48018.1.2.2 (iso.2.840.48018.1.2.2) (MS KRB5 - Microsoft
               Kerberos 5)
              OID: 1.2.840.113554.1.2.2 (iso.2.840.113554.1.2.2) (KRB5 - Kerberos 5
              )
              OID: 1.2.840.113554.1.2.2.3 (iso.2.840.113554.1.2.2.3) (KRB5 - Kerber
              os 5 - User to User)
              OID: 1.3.6.1.4.1.311.2.2.10 (SNMPv2-SMI::enterprises.311.2.2.10) (NTL
              MSSP - Microsoft NTLM Security Support Provider)
            mechListMIC: w2003final$@WIN2K3.THINKER.LOCAL
```

Figure 8.7: Negotiate Protocol Reply



```
No.     Time        Source                Destination           Proto
col Info
    224 396.149513  192.168.3.129         192.168.3.3           KRB5
    AS-REQ

Frame 224 (371 bytes on wire, 371 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:0c:29:ff:62:81
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.3 (192.168.3.3)
User Datagram Protocol, Src Port: 1047 (1047), Dst Port: kerberos (88
)
Kerberos AS-REQ
  Pvno: 5
  MSG Type: AS-REQ (10)
  padata: PA-ENC-TIMESTAMP PA-PAC-REQUEST
  KDC_REQ_BODY
    Padding: 0
    KDCOptions: 40810010 (Forwardable, Renewable, Canonicalize, Renewable
     OK)
    Client Name (Principal): administrator
    Realm: WIN2K3.THINKER.LOCAL
    Server Name (Service and Instance): krbtgt WIN2K3.THINKER.LOCAL
    till: 2037-09-13 02:48:05 (Z)
    rtime: 2037-09-13 02:48:05 (Z)
    Nonce: 301462589
    Encryption Types: rc4-hmac rc4-hmac-old rc4-md4 des-cbc-md5 des-cbc-c
    rc rc4-hmac-exp rc4-hmac-old-exp
    HostAddresses: WINXP1<20>

No.     Time        Source                Destination           Proto
col Info
    225 396.152527  192.168.3.3           192.168.3.129         KRB5
    AS-REP

Frame 225 (1447 bytes on wire, 1447 bytes captured)
Ethernet II, Src: 00:0c:29:ff:62:81, Dst: 00:0c:29:c7:09:32
Internet Protocol, Src Addr: 192.168.3.3 (192.168.3.3), Dst Addr: 192
.168.3.129 (192.168.3.129)
User Datagram Protocol, Src Port: kerberos (88), Dst Port: 1047 (1047
)
Kerberos AS-REP
  Pvno: 5
  MSG Type: AS-REP (11)
  Client Realm: WIN2K3.THINKER.LOCAL
  Client Name (Principal): administrator
  Ticket
    Tkt-vno: 5
    Realm: WIN2K3.THINKER.LOCAL
```

Figure 8.8: Initial Kerberos exchange (AS-REQ).

8.3.4 Kerberos Login

Kerberos is a separate networking authentication protocol entirely, and Figure 8.8 shows the client requesting a Ticket Granting Ticket (TGT), allowing it to request other tickets. It does so by sending the current time, encrypted with the user's password (this is the pre-auth data). Of interesting note in this TGT is the HostAddresss (network addresses where this TGT may be validly used) is given as a NetBIOS Name, not the usual IP address! The TGT allows the client to request tickets for services, which in this case is passed to the CIFS server form of a Session Setup request (shown in Figure 8.9), wrapped in SPNEGO and GSSAPI. Note the server name as it appears in the ticket: `cifs/win2003.win2k.thinker.local`.

8.3.5 DCE-RPC over CIFS

The CIFS connection in this case is only made for the purpose of establishing a transport for DCE-RPC, and Figure 8.10 shows the client's first request - looking again for basic information about the domain. Of particular note is the `LsarQueryInformationPolicy2` reply - the presence of this reply (as opposed to a fault indicating an unknown operation) indicates to the client that this is indeed an 'Active Directory' domain. The information supplied is actually very much like that found in the CLDAP reply - an exploded view of this packet is in Figure 8.11.

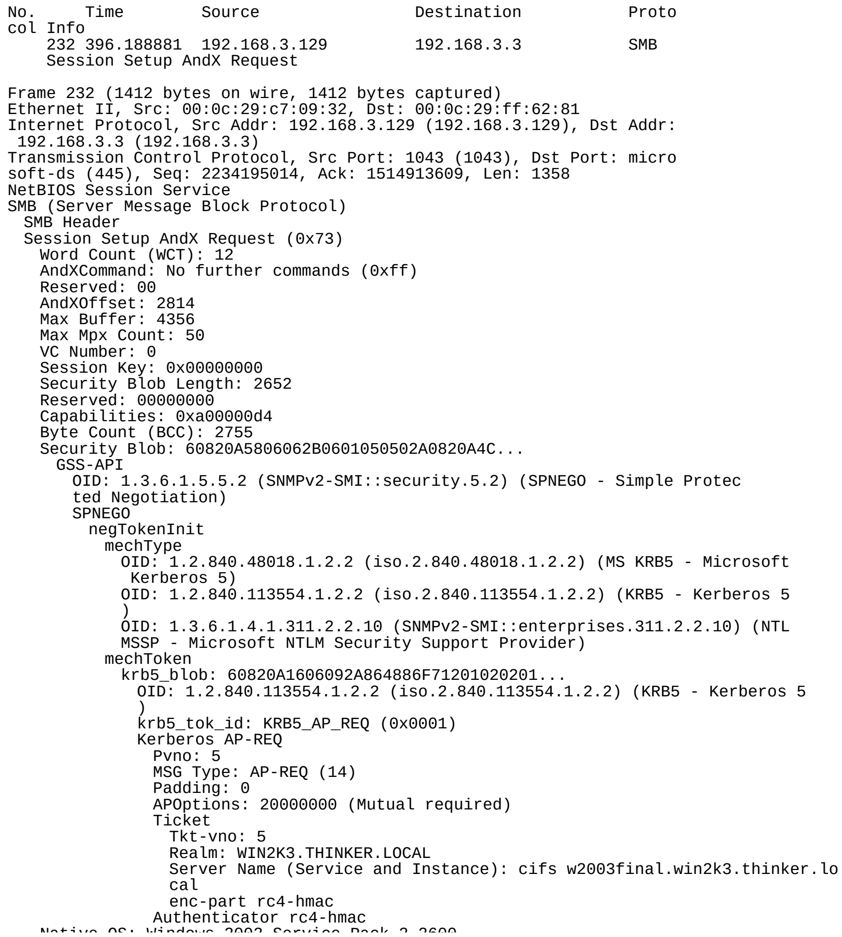
From this point, a stream of requests are made, starting with the `SAMR` pipe, to create the new machine account. Shown in summary in Figure 8.12, the end result is a new machine account, with a password set. The `SamrGetUserDomainPasswordInformation` call is to enquire of the minimum password length, so that the subsequent `SamrSetInformationUser2` request can set a sufficiently long password.

8.4 Modifying the servicePrincipalNames

Up until this point, the join process is very much like that of a Samba or NT4 server, and could be implemented without many changes to an existing code-base. However, the client soon changes the problem drastically, by making a call to the LDAP server, which it presumes to be operating on the same Domain Controller. Combined with yet more DCE-RPC calls, this time to the DRSUAPI (Directory Replication Service) endpoint, the client uses 'standard' LDAP[4] to modify the Kerberos service principal names associated with the machine account.

[3] A point that will become very, very clear at the end of the 1300 packets in this join / domain logon sequence!

[4] Standard in the sense of being TCP based (not UDP, as CLDAP is) and being largely conforment with the LDAPv3 specifications[60].



```
No.     Time        Source                Destination           Proto
col Info
    232 396.188881  192.168.3.129         192.168.3.3           SMB
    Session Setup AndX Request

Frame 232 (1412 bytes on wire, 1412 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:0c:29:ff:62:81
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.3 (192.168.3.3)
Transmission Control Protocol, Src Port: 1043 (1043), Dst Port: micro
soft-ds (445), Seq: 2234195014, Ack: 1514913609, Len: 1358
NetBIOS Session Service
SMB (Server Message Block Protocol)
  SMB Header
  Session Setup AndX Request (0x73)
    Word Count (WCT): 12
    AndXCommand: No further commands (0xff)
    Reserved: 00
    AndXOffset: 2814
    Max Buffer: 4356
    Max Mpx Count: 50
    VC Number: 0
    Session Key: 0x00000000
    Security Blob Length: 2652
    Reserved: 00000000
    Capabilities: 0xa00000d4
    Byte Count (BCC): 2755
    Security Blob: 60820A5806062B0601050502A0820A4C...
      GSS-API
        OID: 1.3.6.1.5.5.2 (SNMPv2-SMI::security.5.2) (SPNEGO - Simple Protec
        ted Negotiation)
        SPNEGO
          negTokenInit
            mechType
              OID: 1.2.840.48018.1.2.2 (iso.2.840.48018.1.2.2) (MS KRB5 - Microsoft
               Kerberos 5)
              OID: 1.2.840.113554.1.2.2 (iso.2.840.113554.1.2.2) (KRB5 - Kerberos 5
              )
              OID: 1.3.6.1.4.1.311.2.2.10 (SNMPv2-SMI::enterprises.311.2.2.10) (NTL
              MSSP - Microsoft NTLM Security Support Provider)
            mechToken
              krb5_blob: 60820A1606092A864886F71201020201...
                OID: 1.2.840.113554.1.2.2 (iso.2.840.113554.1.2.2) (KRB5 - Kerberos 5
                )
                krb5_tok_id: KRB5_AP_REQ (0x0001)
                Kerberos AP-REQ
                  Pvno: 5
                  MSG Type: AP-REQ (14)
                  Padding: 0
                  APOptions: 20000000 (Mutual required)
                  Ticket
                    Tkt-vno: 5
                    Realm: WIN2K3.THINKER.LOCAL
                    Server Name (Service and Instance): cifs w2003final.win2k3.thinker.lo
                    cal
                    enc-part rc4-hmac
                  Authenticator rc4-hmac
    Native OS: Windows 2002 Service Pack 2 2600
```

Figure 8.9: Session Setup, using the Kerberos ticket, wrapped in GSSAPI and SPNEGO.

```
No.     Time        Source                Destination           Proto
col Info
    305 397.110013  192.168.3.129         192.168.3.3           SMB
    Tree Connect AndX Request, Path: \\W2003FINAL.WIN2K3.THINKER.LOCA
L\IPC$
    306 397.110452  192.168.3.3           192.168.3.129         SMB
    Tree Connect AndX Response
    307 397.140693  192.168.3.129         192.168.3.3           SMB
    NT Create AndX Request, Path: \lsarpc
    308 397.141827  192.168.3.3           192.168.3.129         SMB
    NT Create AndX Response, FID: 0x4000
    309 397.157207  192.168.3.129         192.168.3.3           DCERP
C   Bind: call_id: 1 UUID: LSA
    312 397.163569  192.168.3.3           192.168.3.129         DCERP
C   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
    313 397.168840  192.168.3.129         192.168.3.3           LSA
    LsarOpenPolicy2 request, \\w2003final.win2k3.thinker.local
    314 397.169793  192.168.3.3           192.168.3.129         LSA
    LsarOpenPolicy2 response
    315 397.172297  192.168.3.129         192.168.3.3           LSA
    LsarQueryInformationPolicy2 request
    316 397.172819  192.168.3.3           192.168.3.129         LSA
    LsarQueryInformationPolicy2 response
    317 397.173373  192.168.3.129         192.168.3.3           LSA
    LsarQueryInformationPolicy request, Primary Domain Information
    318 397.177576  192.168.3.3           192.168.3.129         LSA
    LsarQueryInformationPolicy response
```

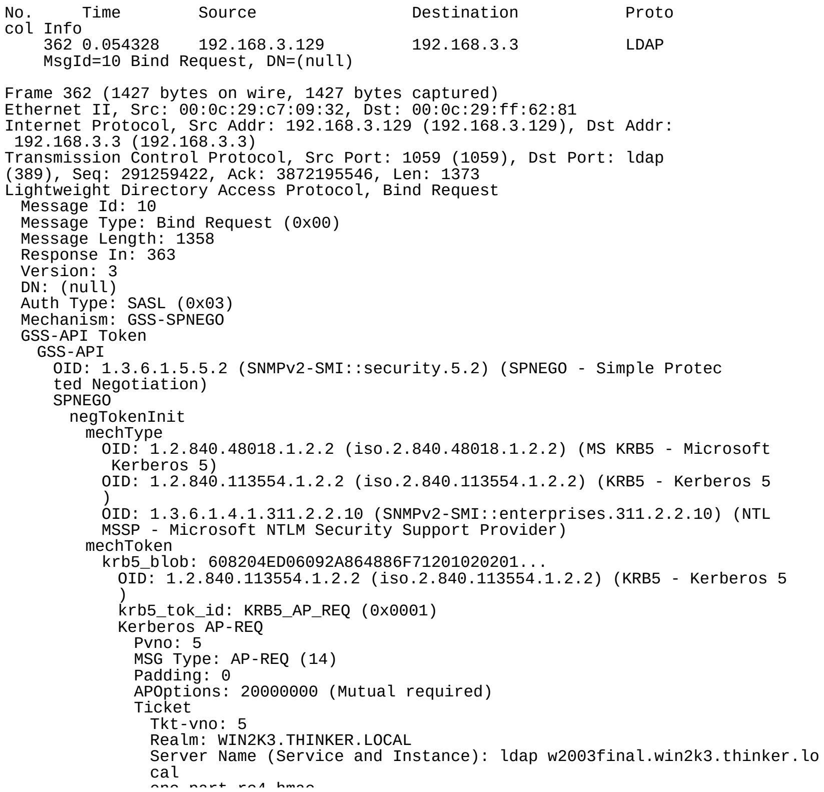
Figure 8.10: The initial RPC requests

```
No.     Time        Source                Destination           Proto
col Info
    316 397.172819  192.168.3.3           192.168.3.129         LSA
    LsarQueryInformationPolicy2 response

Frame 316 (362 bytes on wire, 362 bytes captured)
Ethernet II, Src: 00:0c:29:ff:62:81, Dst: 00:0c:29:c7:09:32
Internet Protocol, Src Addr: 192.168.3.3 (192.168.3.3), Dst Addr: 192
.168.3.129 (192.168.3.129)
Transmission Control Protocol, Src Port: microsoft-ds (445), Dst Port
: 1054 (1054), Seq: 3240586603, Ack: 1847857587, Len: 308
NetBIOS Session Service
SMB (Server Message Block Protocol)
SMB Pipe Protocol
DCE RPC
Microsoft Local Security Architecture, LsarQueryInformationPolicy2
  Operation: LsarQueryInformationPolicy2 (46)
  POLICY_INFORMATION pointer: info
    Referent ID: 0x00020000
    POLICY INFO
      Level: 12
      POLICY_DNS_DOMAIN_INFO:
        Domain: WIN2K3THINK
        FQDN: win2k3.thinker.local
        Forest: win2k3.thinker.local
        GUID: 5de22722-2016-4924-b8fa-6e23bcb37987
        SID pointer:
          SID pointer
            Referent ID: 0x00020010
            Count: 4
            Domain SID: S-1-5-21-3048156945-3961193616-3706469200 (WIN2K3THINK)
  Return code: STATUS_SUCCESS (0x00000000)
```

Figure 8.11: LsarQueryInformationPolicy2 Reply

```
No.     Time        Source                Destination           Proto
col Info
    319 397.259514  192.168.3.129         192.168.3.3           SMB
    NT Create AndX Request, Path: \samr
    320 397.260329  192.168.3.3           192.168.3.129         SMB
    NT Create AndX Response, FID: 0x4001
    321 397.261567  192.168.3.129         192.168.3.3           DCERP
C   Bind: call_id: 1 UUID: SAMR
    324 397.263096  192.168.3.3           192.168.3.129         DCERP
C   Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
    325 397.263773  192.168.3.129         192.168.3.3           SAMR
    SamrConnect5 request
    326 397.264451  192.168.3.3           192.168.3.129         SAMR
    SamrConnect5 response
    327 397.265250  192.168.3.129         192.168.3.3           SAMR
    SamrEnumerateDomainsInSamServer request
    328 397.265673  192.168.3.3           192.168.3.129         SAMR
    SamrEnumerateDomainsInSamServer response
    329 397.266144  192.168.3.129         192.168.3.3           SAMR
    SamrLookupDomainInSamServer request
    330 397.267266  192.168.3.3           192.168.3.129         SAMR
    SamrLookupDomainInSamServer response
    331 397.267991  192.168.3.129         192.168.3.3           SAMR
    SamrOpenDomain request, S-1-5-21-3048156945-3961193616-3706469200
 (WIN2K3THINK)
    332 397.269400  192.168.3.3           192.168.3.129         SAMR
    SamrOpenDomain response
    333 397.274390  192.168.3.129         192.168.3.3           SAMR
    SamrCreateUser2InDomain request
    335 397.532148  192.168.3.3           192.168.3.129         SAMR
    SamrCreateUser2InDomain response
    336 397.554116  192.168.3.129         192.168.3.3           SAMR
    SamrQueryInformationUser request, level 16
    337 397.555634  192.168.3.3           192.168.3.129         SAMR
    SamrQueryInformationUser response
    338 397.556988  192.168.3.129         192.168.3.3           SAMR
    SamrGetUserDomainPasswordInformation request
    339 397.557967  192.168.3.3           192.168.3.129         SAMR
    SamrGetUserDomainPasswordInformation response
    340 397.595767  192.168.3.129         192.168.3.3           SAMR
    SamrSetInformationUser2 request, level 25[Long frame (754 bytes)]
    341 397.663441  192.168.3.3           192.168.3.129         SAMR
    SamrSetInformationUser2 response
    342 397.665644  192.168.3.129         192.168.3.3           SAMR
    SamrCloseHandle request, CreateUser2 handle
    343 397.666361  192.168.3.3           192.168.3.129         SAMR
    SamrCloseHandle response
    344 397.667149  192.168.3.129         192.168.3.3           SAMR
    SamrCloseHandle request, OpenDomain(S-1-5-21-3048156945-396119361
6-3706469200 (WIN2K3THINK))
    345 397.667833  192.168.3.3           192.168.3.129         SAMR
    SamrCloseHandle response
    346 397.668356  192.168.3.129         192.168.3.3           SAMR
    SamrCloseHandle request
    347 397.677871  192.168.3.3           192.168.3.129         SAMR
    SamrCloseHandle response
    348 397.680342  192.168.3.129         192.168.3.3           SMB
    Close Request, FID: 0x4001
    349 397.681108  192.168.3.3           192.168.3.129         SMB
    Close Response
```

Figure 8.12: Account creation over SAMR



```
No.     Time        Source                Destination           Proto
col Info
    362 0.054328    192.168.3.129         192.168.3.3           LDAP
    MsgId=10 Bind Request, DN=(null)

Frame 362 (1427 bytes on wire, 1427 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:0c:29:ff:62:81
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.3 (192.168.3.3)
Transmission Control Protocol, Src Port: 1059 (1059), Dst Port: ldap
(389), Seq: 291259422, Ack: 3872195546, Len: 1373
Lightweight Directory Access Protocol, Bind Request
  Message Id: 10
  Message Type: Bind Request (0x00)
  Message Length: 1358
  Response In: 363
  Version: 3
  DN: (null)
  Auth Type: SASL (0x03)
  Mechanism: GSS-SPNEGO
  GSS-API Token
    GSS-API
      OID: 1.3.6.1.5.5.2 (SNMPv2-SMI::security.5.2) (SPNEGO - Simple Protec
      ted Negotiation)
      SPNEGO
        negTokenInit
          mechType
            OID: 1.2.840.48018.1.2.2 (iso.2.840.48018.1.2.2) (MS KRB5 - Microsoft
             Kerberos 5)
            OID: 1.2.840.113554.1.2.2 (iso.2.840.113554.1.2.2) (KRB5 - Kerberos 5
            )
            OID: 1.3.6.1.4.1.311.2.2.10 (SNMPv2-SMI::enterprises.311.2.2.10) (NTL
            MSSP - Microsoft NTLM Security Support Provider)
          mechToken
            krb5_blob: 608204ED06092A864886F71201020201...
              OID: 1.2.840.113554.1.2.2 (iso.2.840.113554.1.2.2) (KRB5 - Kerberos 5
              )
              krb5_tok_id: KRB5_AP_REQ (0x0001)
              Kerberos AP-REQ
                Pvno: 5
                MSG Type: AP-REQ (14)
                Padding: 0
                APOptions: 20000000 (Mutual required)
                Ticket
                  Tkt-vno: 5
                  Realm: WIN2K3.THINKER.LOCAL
                  Server Name (Service and Instance): ldap w2003final.win2k3.thinker.lo
                  cal
                  enc-part rc4-hmac
```

Figure 8.13: The SASL/SPNEGO/Kerberos LDAP bind request.

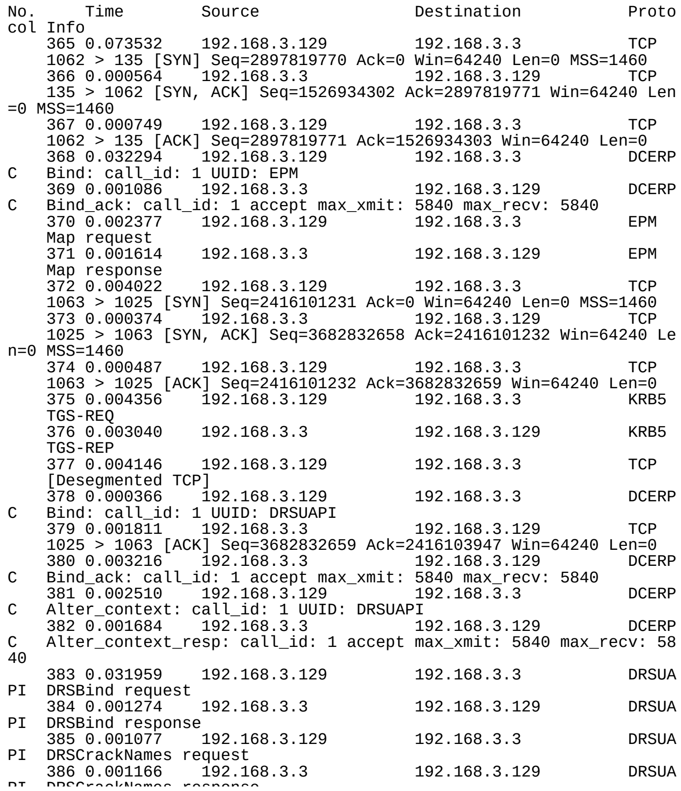
This series of calls is interesting for a number of reasons, but particularly because of the implementation challenges it presents:

8.4.1 LDAP Bind

The LDAP protocol consists of a connect, and optionally a bind (authenticate request) as a particular user. In the case of Active Directory, the client uses an authenticated bind, with the credentials of the user joining the domain. In this case, shown in Figure 8.13, the bind is performed using Kerberos, which is wrapped rather oddly in both SASL (described in 7.1) and SPNEGO (7.2.1).

8.4.2 DCE-RPC over TCP

After confirming that the LDAP server is contactable, and a login has been performed, figure 8.14 shows the client making a DCERPC End Point Mapper (EPM) call to locate the DRSUAPI pipe, and performing a similar SPENGO/KRB5 bind (login). This sets up an encrypted connection on that pipe, which prevents the data payload from being revealed. However, the function num-



```
No.     Time        Source                Destination           Proto
col Info
    365 0.073532    192.168.3.129         192.168.3.3           TCP
    1062 > 135 [SYN] Seq=2897819770 Ack=0 Win=64240 Len=0 MSS=1460
    366 0.000564    192.168.3.3           192.168.3.129         TCP
    135 > 1062 [SYN, ACK] Seq=1526934302 Ack=2897819771 Win=64240 Len
=0 MSS=1460
    367 0.000749    192.168.3.129         192.168.3.3           TCP
    1062 > 135 [ACK] Seq=2897819771 Ack=1526934303 Win=64240 Len=0
    368 0.032294    192.168.3.129         192.168.3.3           DCERP
C   Bind: call_id: 1 UUID: EPM
    369 0.001086    192.168.3.3           192.168.3.129         DCERP
C   Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
    370 0.002377    192.168.3.129         192.168.3.3           EPM
    Map request
    371 0.001614    192.168.3.3           192.168.3.129         EPM
    Map response
    372 0.004022    192.168.3.129         192.168.3.3           TCP
    1063 > 1025 [SYN] Seq=2416101231 Ack=0 Win=64240 Len=0 MSS=1460
    373 0.000374    192.168.3.3           192.168.3.129         TCP
    1025 > 1063 [SYN, ACK] Seq=3682832658 Ack=2416101232 Win=64240 Le
n=0 MSS=1460
    374 0.000487    192.168.3.129         192.168.3.3           TCP
    1063 > 1025 [ACK] Seq=2416101232 Ack=3682832659 Win=64240 Len=0
    375 0.004356    192.168.3.129         192.168.3.3           KRB5
    TGS-REQ
    376 0.003040    192.168.3.3           192.168.3.129         KRB5
    TGS-REP
    377 0.004146    192.168.3.129         192.168.3.3           TCP
    [Desegmented TCP]
    378 0.000366    192.168.3.129         192.168.3.3           DCERP
C   Bind: call_id: 1 UUID: DRSUAPI
    379 0.001811    192.168.3.3           192.168.3.129         TCP
    1025 > 1063 [ACK] Seq=3682832659 Ack=2416103947 Win=64240 Len=0
    380 0.003216    192.168.3.3           192.168.3.129         DCERP
C   Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
    381 0.002510    192.168.3.129         192.168.3.3           DCERP
C   Alter_context: call_id: 1 UUID: DRSUAPI
    382 0.001684    192.168.3.3           192.168.3.129         DCERP
C   Alter_context_resp: call_id: 1 accept max_xmit: 5840 max_recv: 58
40
    383 0.031959    192.168.3.129         192.168.3.3           DRSUA
PI  DRSBind request
    384 0.001274    192.168.3.3           192.168.3.129         DRSUA
PI  DRSBind response
    385 0.001077    192.168.3.129         192.168.3.3           DRSUA
PI  DRSCrackNames request
    386 0.001166    192.168.3.3           192.168.3.129         DRSUA
PI  DRSCrackNames response
```

Figure 8.14: DCE-RPC over TCP

bers are still visible, and we can determine a little without need to decrypt the entire connection.

We can see from the exposed function numbers that a call to DsCrackNames is made, which we can assume from the context makes a conversion between a machine account name[5] and its LDAP DN[6]. We know it must return the DN, because the LDAP search in packet 391 (in Figure 8.15) uses the exact name of the machine[7].

8.4.3 LDAP search and modification

The next step is to search LDAP for that exact DN, looking for an existing DNS host name, and any existing servicePrincipalName. Figure 8.16

[5] This is known, because it was just created, and a password was just set on it.

[6] A DN, or Distinguished Name, is the unique location of an entry in the directory tree. The DN in this case is `CN=WINXP1,CN=Computers,DC=win2k3,DC=thinker,DC=local`.

[7] This whole procedure could be considered pointless, as a similar search for the machine name could be performed directly in LDAP. However, it is clear from the MSDN documentation[34] that `DsCrackNames` provides a simple API for this purpose.

```
No.     Time        Source                Destination           Proto
col Info
    391 0.001030    192.168.3.129         192.168.3.3           LDAP
    MsgId=11 Search Request, Base DN=CN=WINXP1,CN=Computers,DC=win2k3
,DC=thinker,DC=local
    392 0.001113    192.168.3.3           192.168.3.129         LDAP
    MsgId=11 Search Entry
    393 0.001443    192.168.3.129         192.168.3.3           LDAP
    MsgId=12 Modify Request
    394 0.006098    192.168.3.3           192.168.3.129         LDAP
    MsgId=12 Modify Result
    395 0.005582    192.168.3.129         192.168.3.3           LDAP
    MsgId=13 Unbind Request
```

Figure 8.15: LDAP Search and Modification

```
No.     Time        Source                Destination           Proto
col Info
    393 0.001443    192.168.3.129         192.168.3.3           LDAP
    MsgId=12 Modify Request

Frame 393 (334 bytes on wire, 334 bytes captured)
Ethernet II, Src: 00:0c:29:c7:09:32, Dst: 00:0c:29:ff:62:81
Internet Protocol, Src Addr: 192.168.3.129 (192.168.3.129), Dst Addr:
 192.168.3.3 (192.168.3.3)
Transmission Control Protocol, Src Port: 1059 (1059), Dst Port: ldap
(389), Seq: 291260984, Ack: 3872195881, Len: 280
Lightweight Directory Access Protocol, Modify Request
  SASL Buffer Length: 276
  GSS-API Token
  Message Id: 12
  Message Type: Modify Request (0x06)
  Message Length: 213
  Response In: 394
  Distinguished Name: CN=WINXP1,CN=Computers,DC=win2k3,DC=thinker,DC=lo
  cal
  Add: DnsHostName
    Value: winxp1.win2k3.thinker.local
  Add: ServicePrincipalName
    Value: HOST/winxp1.win2k3.thinker.local
    Value: HOST/WINXP1
```

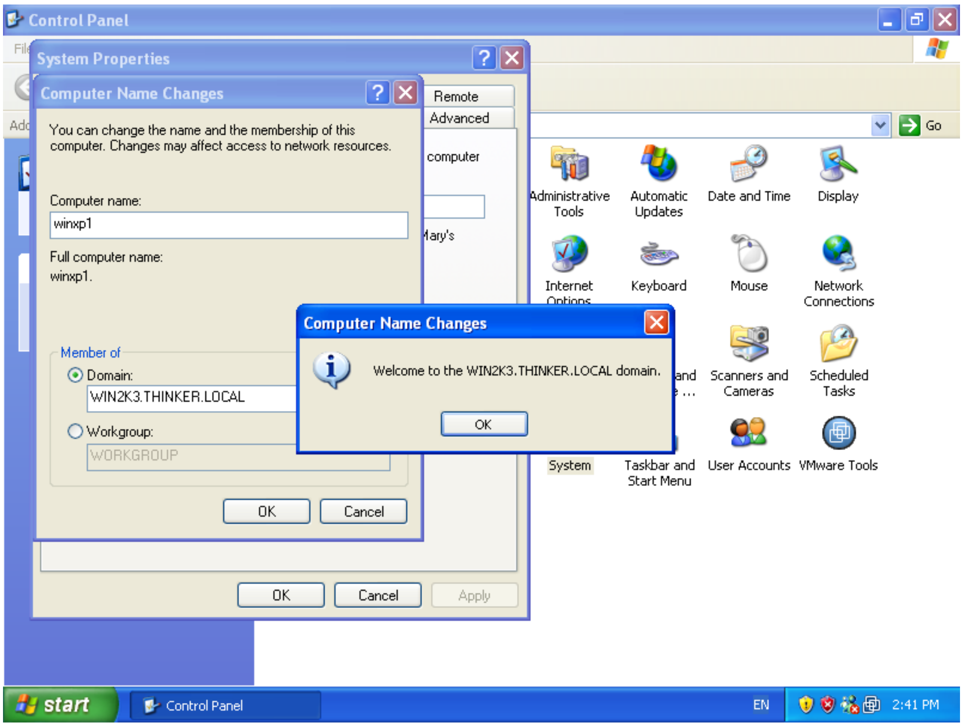
Figure 8.16: Adding new servicePrincipalName attributes

shows the subsequent modify request, where new servicePrincipalName attributes are added, per the standard scheme for a member server.

8.5 Information Discovery

At this point, the machine is joined to the domain, and few further modifications of the server appear to occur. Instead, the machine now takes the opportunity to discover certain information about the domain that it joined. This process is incredibly verbose, but consists primarily of RPC queries, DNS lookups and LDAP searches.

It is not worthwhile to analyse each call in detail at this point, as our existing knowledge provides sufficient work for analysis and implementation as it is. That said, an examination of the packets shows new calls, but no new technologies.



Computer Name Changes

You can change the name and the membership of this computer. Changes may affect access to network resources.

Computer name: winxp1

Full computer name: winxp1.

Member of

● Domain: WIN2K3.THINKER.LOCAL

○ Workgroup: WORKGROUP

Computer Name Changes

Welcome to the WIN2K3.THINKER.LOCAL domain.

OK

Figure 8.17: Successfully joined to the domain

8.6 Success!

The join process concludes, and the user is informed of the join's success, as in figure 8.17.

Chapter 9

The Challenge of Active Directory

The challenge of Active Directory is not the individual components - each can be broken down into simple problems, most of which are not particularly novel in computer science. Instead, the challenge in achieving Active Directory compatibility is that of integration. Each and every one of the above protocols and authentications schemes all work together on a common basis. As is clear from the join process - the LDAP server is expected to contain the results of the SAMR call, as soon as that call is made. Likewise, the Kerberos server is expected to honour both the passwords and the servicePrincipalNames set by SAMR and LDAP. While there are numerous views on the Data model, Microsoft's own documentation, and the behaviour on the network, make it clear that the back-end is eventually a single Jet[1] Database[31].

As such, the goal of Active Directory compatibility must be achieved by integration, and as such the Samba4 implementation must strongly reflect that.

[1] Jet is the same database technology used behind Microsoft Access, Microsoft Visual Basic and Microsoft Visual C++

Part II

Implementation

Chapter 10

Open Source Building Blocks

10.1 Samba

Samba provides Windows networking services, on a Unix-like platform. These services range from simple file and printer sharing, to full management of a NT-style domain. All of these services are provided in the Samba package, which is itself distributed under the Free Software Foundation's[13] General Public Licence (GPL).[14]

10.1.1 History of Samba

Started by Andrew Tridgell, during his PhD studies [53], Samba quietly evolved over the past 12 years from a barely functional prototype, used to communicate between a DOS Pathworks client and a Sun server, into a solid file and print server for Windows clients, maintained by a team of over 30 international developers, 12 of which are active at any one time.

Samba 2.0

After years of 1.x and in particular 1.9.x releases, Samba 2.0 brought new levels of protocol completeness to the Samba project, and initial support for becoming a domain member.

Samba 2.2

Samba 2.2 brought the first implementation of a Domain Controller to Samba's stable series, and provided a solid domain member platform with the introduction of the `winbindd` daemon in Samba 2.2.3.

Samba 3.0

With the introduction of Samba 3.0, Samba finally used the Unicode character representation when talking to network clients, solving many issues in non-English environments. Samba 3.0 also featured a vastly improved domain controller, and support for being a client of Active Directory.

10.1.2 Samba as a DC

Samba 2.2[2] and in particular Samba 3.0 grew to include the ability to be an NT4 compatible domain controller, a functionality that even allows Samba to 'take over' an existing Windows network[51]. This has allowed many sites to remove Windows servers entirely from their networks.

Because Samba 3.0 implements the full requirements of an NT4 DC, it can be used to host some of the legacy parts of the protocol, not yet found in Samba4 - in particular, NetBIOS name registration and NETLOGON requests. Patches have been proposed (and some already accepted) to allow this piece of Samba3 infrastructure to handle these roles, in the Samba4 framework.

10.1.3 Samba as a Active Directory domain member

Samba 3.0 release [45] has the ability to be a member of an Active Directory domain, and as such has an implementation of a particular form of AD client. This client uses Kerberos for authentication, and used DCE-RPC and LDAP to query user and group information from the DCs.

10.1.4 Samba 3.0 Active Directory DC research

As part of a internship project known as Blue Directory, students and supervisors at IBM's Linux Technology Center spent a lot of time researching the problem space around Active Directory, but as described by McDonough [30], they kept hitting up against limitations in the available technology. Their research work has been rapidly superseded by the Samba4 effort, but their input showed what would be possible with the proper infrastructure.

10.2 Heimdal Kerberos

Heimdal [18, 62] is an Open Source implementation of the Kerberos protocol. Created outside the USA due to export controls on strong encryption [7, 37], it has been developed independently of the well-known MIT distribution[38]. The Heimdal source code is well tested, and quite easy to modify. The presence of the HDB back-end interface (not found in the MIT distribution) is what made Heimdal the clear choice for this integration effort.

Another aspect that makes Heimdal a key building block in this effort has been the active participation of key Heimdal developers in our branch of the Heimdal source [23].

10.2.1 HDB Back-end

Within Heimdal, there is an abstraction layer that separates the password database from the rest of the Kerberos implementation. In the unmodified code, this allows the administrator to select between an LDAP back-end, and a simple key-value database. It is this interface that this project will extend, with a new 'ldb' back-end to be provided.

10.2.2 Heimdal/Samba Integration

Another feature of current Heimdal snapshots is support for integration with Samba 3.0. By using Samba 3.0's password entries in the LDAP database, Heimdal snapshots can use the `sambaNTPassword` attribute as an `arcfour-hmac-md5` Kerberos key. This integration work not only opened up valuable communication channels between Samba and Heimdal developers, it provided hands-on experience in hdb module development.

10.3 `clapd`

`clapd` is a simple Connectionless LDAP daemon, written as part of the IBM research effort[30] and designed to answer the basic requests that a Windows client makes over connectionless LDAP. At present, it is functional to the extent required for my domain join test, but has failed for others. It will need to be rewritten and properly integrated into the Samba4 system.

10.4 BIND

An Active Directory domain is strongly based on a DNS domain, particularly due to the tight integration between Kerberos and DNS, and the fact that this allows a move to a hierarchical name space. No modifications have been required to the BIND software, and only the installation of configuration files is required.

In the future, changes to BIND will be required to support Microsoft's dynamic DNS update scheme.

Chapter 11

Samba4 Status

Samba version 4 is an ongoing research project of the Samba Team, and had made significant headway into this problem space before I even proposed my thesis topic. It has grown up in a very modular style, and with a much cleaner code-base than Samba 3.0. The core development on Samba4 has been by Andrew Tridgell, Stefan Metzmacher and myself, with contributions from many others from time to time.

While there is far more to Samba4 than these subsystems, the AD emulation work hits on these in particular:

11.1 LDB

LDB is described as a 'LDAP like Database'. Designed to avoid giving Samba4 a dependency on OpenLDAP, ldb implements an 'LDAP like' API and data format. LDB includes an abstraction layer that allows it to be backed onto LDAP, or onto a local flat-file database, in the format of a TDB. Therefore, a Samba4 installation can remain self-contained, without demanding the notoriously complex task of setting up an external LDAP server.

LDB is available as a shared library, and is licenced under the LGPL. This allows other applications to link against our database back end.

11.2 GENSEC

Intended to replicate SSPI [32] on Windows in ubiquitous use, GENSEC allows all aspects of Samba4 to use the same implementation of our core authentication protocols. The interface is completely generic, and on starting this thesis it contained support only for NTLMSSP, the development of which had been brought forward from Samba 3.0. The addition of Kerberos support to this interface is one of the core objectives in this thesis.

11.3 PIDL - Midl replacement for Samba

One of the biggest achievements in Samba4's design and development is the PIDL[10] IDL compiler. Known as PIDL because it is a Perl compiler of the Interface Definition Language, this comparatively small collection of Perl modules and handling scripts converts the standard IDL file format into C code for use inside Samba (On the Microsoft platform a program known as `midl[36]` does the same thing). Attempted, but not completed, in the past, this contrasts greatly with the approach in Samba3 - that of manual parsing of packets, with little regard to the inbuilt rules of the Network Data Representation (NDR) format.

11.4 Echo Pipe

In development of a generic DCE-RPC infrastructure, it is often helpful to have some way to 'define' the problem. Many aspects of DCE-RPC are only exercised in particular circumstances. By defining our own DCE-RPC pipe, it becomes possible to create overly large request and replies, for example. Similarly, by working with a protocol that is simple, and with a public definition, we can expose bugs in the layers that handle simple parsing of the packets.

The particular success of the echo pipe in Samba development is the Windows client and server. By running the echo server on a Microsoft platform, questions such as 'how much data can I pass in a DCE-RPC packet' can be answered, without first contriving to find a piece of Microsoft software that creates large RPC replies. Likewise, the Windows echo client allows great opportunities for testing authentication technologies.

11.5 Domain Join

At the commencement of this thesis, Tridgell [54] had recently claimed success in joining a WinXP machine to the Samba4 domain controller. Naturally, this is a process that must be repeated by others, and I took this on as one of the first tasks in the thesis. While a packet-by-packet analysis of the join is not warranted, it is worth noting two particular characteristics:

11.5.1 An Upgraded NT4 Join

In replicating the domain join (with a WinXP client), I found that the domain join process would only complete when using the domain's 'short' or NetBIOS name. Use of the long (DNS, and Kerberos realm) name would result in calls the (unimplemented) LDAP server, and failure. In the short-name case, the domain join proceeds using just SAMR calls, much the same way as that shown in Section 8.3.5.

11.5.2 A more AD-like Logon

Once the domain join has completed, the logon phase is more 'AD like'. That is, DNS lookups are performed on the long name, and the CLDAP reply is expected, likewise the End Point Mapper (EPM), (not found in NT4) is contacted and newer `netlogon` calls are made, negotiating security mechanisms such as 128-bit session keys.[1]

[1] The challenge of implementing these new extensions to `netlogon` is detailed in Section A.2.

Chapter 12

Adding Kerberos

The addition of Kerberos to the Samba4 implementation forms the experimental basis for this thesis. While the domain join has been demonstrated using the older NTLM technologies (no mean feat in itself), the addition of Kerberos remains critical to any real claim to implement AD.

12.1 Heimdal Integration

As has already been described, the Heimdal implementation of Kerberos includes a database abstraction layer, known as hdb. The challenge in this experiment is to modify Heimdal in such as way that it can use the Samba4 ldb as a password database. To allow easy development, a copy of the Heimdal development snapshots were imported into the Samba 'lorikeet' subversion repository. This allowed easy modification and version control, separate from the original Heimdal authors. Care was taken to liaise with Heimdal's developers to reassure them of our intentions, and that we did not intend this to be a long-term fork of the Heimdal code-base.

12.1.1 A new hdb-ldb for Heimdal

The obvious first step in the process was to take the hdb-ldap, long established Heimdal component, and modify it to call ldb (the Samba4 'ldap like' database API) rather than LDAP. This work was initially performed by Stefan Metzmacher, and extended by myself, and essentially consists of a semantic mapping exercise - mapping ldb attributes into the form defined as the Heimdal hdb interface. Because Microsoft extended Kerberos with the `servicePrincipalName` concept (a principal may be resolved by many names), this needed to be carefully addressed in LDB.

This process was iterative, and at each stage of development tests were performed - starting with a Unix `kinit` and later by using a WinXP client, as part of testing on the entire domain join operation. This showed up defects, and at each stage a more complete semantic mapping was devised.

12.1.2 Heimdal Structural Changes

A few aspects of the hdb-ldb development created problems, due to the structure and assumptions in the original Heimdal code. In particular, hdb-ldb is unique in Heimdal in that it may contain plain-text passwords, not the hashed encryption keys. Storing the plain-text password is required when the 'store password with reversible encryption' flag is set, and doing so by default has allowed easier initial development. In this case, we 'hash' the passwords on the fly, but we can also store the pre-hashed password if the plain-text is not available. These requirements required me to perform a minor code restructure, which has now been included by Heimdal's developers into current snapshots.

12.1.3 No PAC at this stage

While the PAC is now well understood, and sample KDC implementations that successfully sign the PAC have been contructedMcDonough [30], PAC support was not implemented: instead the Samba server was modified to accept Kerberos packets without the PAC. (The client does not process the PAC in the initial use case, so this complexity was deferred).

12.2 Adding Kerberos to Samba4

By this stage, Samba4 had gained the new GENSEC security subsystem, and it was from this base that Kerberos client and server support was added. In particular, Stefan Metzmacher added support for parsing (but not yet verifying the signatures on) the PAC, and we both worked on the basic infrastructure required to produce and accept a Kerberos ticket. Despite being a re-implementation of functionality already working in Samba 3.0, the new GENSEC system needed work to cope with how Kerberos functions, and the Kerberos code was again hooked into mechanisms such as SMB signing.

This would allow an attempted domain join (the target of this experiment) to use Kerberos.

12.3 Testing the Experiment - the Kerberos Domain Join

Now that Kerberos has been added to Samba4, and the Heimdal Kerberos distribution has been modified to read information from ldb, the changes can be assessed.

12.3.1 'Long name' domain join

In my testing the of a Kerberos domain join, it quickly becomes clear that the domain would not use Kerberos for the join if the 'short' (NetBIOS) domain name was specified. While it is no more painful to enter the long name, the point made in Section 11.5.1 still holds - such a domain join will contact LDAP and DRSUAPI, and fail. Fortunately LDAP server support is progressing rapidly, as is the implementation of DRSUAPI (and the associated Kerberos security layers), but these were not ready at the technical completion of work for the thesis.

12.3.2 Kerberos in CIFS

Adding correct Kerberos (and SPNEGO) support into CIFS was the primary challenge in the Samba4 code-base, and in this area the project was very much a success. In this case, the keytab was manually exported from Heimdal, and the WinXP client successfully obtained a ticket, and used that ticket to log into the Samba4 server. We also know that the session key logic is correct, because both SMB signing and the password set occur correctly.

12.3.3 Failing at LDAP

The failure in the domain join process occurs as the client attempts to contact the LDAP server, where it wishes to perform the process described in Section 8.4. As mentioned above, this blocking point should be removed in near-future development, but this marks the conclusion of the development for this thesis.

12.3.4 Trying the short name

After completing the 'failed' testing with long-name domain joins, comparison tests were run - to confirm the status of the 'before' case, which was expected to easily be triggered by simply using the 'short' or NetBIOS domain name. Indeed, this domain join proceeds with NTLMSSP, but what is more interesting is what happens after the client reboots, having sucessfully joined the domain. It turns out that the client clues in on Kerberos in the meantime, and proceeds to make a sucessful Kerberos connection to the Samba4 server, using the machine account! Unfortunately, other issues with current Samba4 prevent the login from proceeding, but these appear unrelated to the Kerberos work (it occurs with Kerberos completely disabled), and is rather an unrelated regression.

12.4 Assessing the Changes

The changes for this particular experiment were not large - they could perhaps better be described as tedious, with details in the infrastructure being brought to the surface. However, the change in the domain join procedure was easily noticed - the client (WinXP) would indeed use Kerberos, and this would succeed to the stage (described in Section 8.4.1) of contacting the LDAP server. Naturally, this failed due to the lack of an integrated LDAP server in the Samba4 suite.

This presented an anticlimax - progress has indeed been made, but if it is used, the domain join demonstrated previously fails. This does not however condemn this experiment as a failure - it simply reminds us that more work needs to be done. *The KDC is indeed operable, and Samba4 can use it for simple CIFS logins.*

Since this work was performed an integrated LDAP server has appeared, and will soon support Kerberos logins itself, and we can again test this procedure ‘one step further’. It is known from Section 8.4.2 of the trace, that we will also need Kerberos encrypted DRSUAPI support. Likewise, the compromised ‘short name’ join looks very promising, sucessfully putting off the Kerberos until the logon stage.

Part III

Appendix

Appendix A

Crypto Challenges

In implementing many of the Active Directory protocols, there have been a number of cryptographic puzzles which we have had to solve. These have involved determining the encryption routine, or in one case the previously unknown encryption keys used between Microsoft's own client and server.

Cryptographic challenges differ greatly from the normal style of network protocol analysis because, until the correct answer is obtained, there is no indication how 'close' a particular solution may be. This puts off many people from the challenge, but also makes the problem that much more rewarding when the answer finally arrives. The following cryptographic challenges were solved by Tridgell and other members of the Samba Team (including myself) over the course of 2004.

A.1 Password Change Mechanisms

It turns out that in the CIFS and associated protocol suites, there are at least 15 different ways to change or set a user's password. Most of these have their own particular cryptographic system, to avoid disclosure of the old or new passwords during the password change.

A.1.1 Encrypt new passwords with old

This classic technique for password changes has been in documented use since the early 90's, where the AppleTalk network protocol encrypted the user's new password with the old password, when the user wished to change their password. Importantly, to ensure that the password change was not tampered with, the old password was also sent, encrypted with the new password. This is documented in 'Inside AppleTalk'[48]. This cross-encryption technique has been duplicated many times in CIFS, with only changes to what form the passwords take (already 'hashed' passwords or the plain-text), and what cryptographic algorithms are used to secure them. As an example, on only the SAMR pipe, we have 4 different password change functions:

samr_ChangePasswordUser Encryption of old and new hashed passwords with DES. Does not give plain-text to the server.

samr_OemChangePasswordUser2 Encryption of new password with old LM hash, using arcfourMantin [29]. Old LM and hashs encrypted with new hashed password.

samr_ChangePasswordUser2 Encryption of new password with old LM and NT hashs, using arcfourMantin [29]. Old LM and NT hashes encrypted with new hashed password.

samr_ChangePasswordUser3 Identical encryption, but return detailed information of reason for password change rejection.

Where plain-text passwords are encrypted in CIFS, they are randomly padded in a large (512 byte) buffer. This technique is described in Rivest and Sherman [43], and referred to as a 'confounder'. This avoids certain chosen plain-text attacks on the system, because the full input into the encryption algorithm is no longer constant. In particular, given the use of arcfour in the cypher step, this avoids encryption of known constant padding bytes, which can be problematic in RC4.

A.1.2 Encrypt new passwords with administrator passwords

While encrypting new passwords with their former values is good for user password changes, remote password setting is also permitted, and this is generally secured with some derivative of the administrator's password. This presented a number of security issues in modern networks, as the scheme chosen in early versions of NT would allow an attack on the administrator's own password![1] Hence, versions of Windows 2000 introduced a confounder, into the session key (integrated in this case with MD5), to ensure that any two password set operations have distinct encryption keys.

A.1.3 Solving the password change puzzles

Like all cryptographic puzzles, the password change puzzle can be restricted by the known inputs and outputs. It is known what the password is before and after a password change, and we know what the server can store in its user database (that is, the NT and LM passwords). Given the small amount of information exchanged in the password change packets,

[1] With the fixed User Session Key in NTLM, any two password set operations would have the same encryption key, until the administrator changed their own password.

we know . Like we see in the authentication chapters, we know the server can only use the NT and LM passwords in the password change. Therefore, the task is reduced to considering what encryption and digest (hash) functions have been used, and in what order.

The process is tedious, but because Microsoft has a few 'favorite' encryption functions (clearly easily accessed from a central library), combinations of DES, arcfour and MD5 are a very suitable first guess. Once one password change or set function is known, may of the others prove to be simple variations, fixing particular issues (such as adding in the new NT password, or adding a confounder). We look at a couple of puzzles in detail:

A.2 Netlogon 128

As attempts to solve the Active Directory DC problem continued, Tridgell led attempts to solve most of the outstanding cryptographic puzzles that stood in the way. This included a new 128 bit security mode for the `netlogon` pipe, and the associated schannel bulk encryption mechanism. This problem, like most of the previous problems, was already partly solved - Samba 3.0 had support for schannel, and Samba 2.2 has supported the basic security mechanisms on `netlogon`. What was unknown is how the new 128 bit security system worked, with an unmodified packet format.

A.2.1 A 'simple' challenge

The 128-bit Netlogon problem was solved with the assistance of a standalone cryptographic challenge. Simply consisting of a few C files and a makefile, the challenge provides a malleable unit test, completely distinct from the complexities in the rest of samba. The programmer attempting the puzzle needs only to tweak a small file, recompile and run, to test a given possible solution.

It was with a challenge set by Tridgell that I solved part of the Netlogon-128 puzzle (much to his pleasant surprise). By expressing the problem so simply, the hints found in the standards drafts were easily applied, and I turned an answer around a matter of minutes.

A.2.2 Related standards

In solving this puzzle, my attention was drawn to the fact that netlogon is disturbingly similar in cryptographic operation to the new Kerberos encryption functions described in Section 6.2. Reading the standards drafts, it became clear that Tridgell's attempt was almost, but not quite the same as one of the cryptographic functions in the draft. This combined with the discovery that other parts of the key setup are simply 'longer' versions of the traditional 56-bit exchange allowed a full implementation of 128 bit Netlogon security.

A.3 LSAKEY

When communicating over the network, it is possible for the entire communication channel to be secured, by negotiation of bulk encryption. However when the whole channel is not secured, the SAMR and LSA pipe still transmit sensitive data, such as when an administrator sets a user's password, or retrieving 'LSA secrets'. To secure this data a session key is negotiated between the peers - and in `ncacn_np` (the most common authentication modal) this is simply inherited from the session key determined for use at the CIFS layer.

However, when the CIFS layer does not exist (such as in `ncacn_ip_tcp` - the direct transport of DCE-RPC over TCP/IP), a different session key must be used. Likewise, it was found during testing that if the DCE-RPC layer performed any authentication, that the inherited session key was no longer valid.

This presented a challenge - to perform correct testing of all DCE-RPC transports, we needed to know how to perform these encrypted operations.

A.3.1 An opening:

In attempting to solve this puzzle, it came to our attention that the LSA set and query secret functions could provide a cryptographic insight into the problem. This function allows storage of an arbitrary piece of data in the server. By setting the secret encrypted with one (known) session key, and retrieving it over `ncacn_ip_tcp` (and therefore the unknown key), properties of the encryption function can be derived by extracting the ciphertext for a known plain-text.

A.3.2 Controlling the session key

In researching this problem, I noticed that in NTLMSSP, the `KEY_EXCHANGE` option allows the network client to chose the session key. At this stage, it was unknown what encryption function was in use, but by choosing a known weak encryption key, such as all-ones (all zeros was not supported), we could analyise the properties of the cyphertext.

A.3.3 Proof that it's a fixed key

One of the first breakthroughs in solving the puzzle was the realisation that, despite changes in session keys, user-names or passwords, the encrypted secret would not change. This was most puzzling, because secrets are typically encrypted with a value shared between the user and server (which implies that it should change with the user's password, even if somehow disconnected from the key exchange mentioned above).

This strongly suggests that the key is some constant value, possibly a 'dummy' value that was always intended to be replaced with a real encryption key (such as happens in `ncacn_np` without additional authentication).

A.3.4 Key-search

An analysis of the (unchanging) cyphertext indicated that the encryption function used indicated that is was probably unchanged from the DES variant used in the `ncacn_np` case. This indicated the need for a key search, for the secret key shared between Microsoft implementations.

In considering the possible secret keys, I suggested that the key was probably not a random value, but more likely an ASCII string used for initialisation.[2] On this basis, Tridgell constructed a systematic DES key-breaking attack, using a parallel DES decryption engine, starting with upper and lower case ASCII values, sorted by the frequency in which they occur in natural words.

Eventually (and this only took a matter of 24 hours of CPU time) the fixed key was found: `"SystemLibraryDTC"`.

A.3.5 Consequences of a fixed key

The use of fixed keys in cryptography is very much frowned upon - it only takes one effort such as the one mounted by the Samba Team to break the security of all the communications 'secured' with that key. However, in this case the security issues are mitigated by the requirements for authentication and at a minimum cryptographic signing of the data stream. As such, the client can be careful not to ask for such secrets across a clear-text channel, and an attacker cannot spoof the requests of the client, because that would break the cryptographic signature. The client should instead use the bulk encryption security of the entire session to secure the transport of these sensitive data items.

[2] Other Microsoft security functions use ASCII strings as dummy values, such as SMB signing, where the signature is set to `"BSRSPYL  "` as an initial value.

Appendix B

Glossary

DES Data Encryption Standard, a US Government encryption standard.

NT Domains Windows NT domains share the information about users, groups and passwords between machines in the domain. The protocols are limited in what information can be stored, and is not extensible. Windows 2000 and Samba both provide "NT Domain" views onto their more complex directory back-ends.

LDAP An Internet standard directory services interface. While LDAP is a protocol specification, the protocol implies the X.500 information model.

LM hash The user's case insensitive, ASCII password, processed with DES, as described in Section 5.1.1 (comparatively weak).

NT hash The user's case sensitive, Unicode password, processed with MD4. Used for NTLM and Kerberos authentication.

NTLM A challenge-response authentication protocol, based on shared secrets (passwords).

NTLM2 The description applied to an NTLMSSP negotiated option, that improves the NTLM security system.

NLTMSSP NTLMSSP is a generic wrapping of the NTLM authentication protocol, encompassing negotiated options and an encryption engine.

NTLMv2 A modified version of NTLM, with a true 128 bit challenge/response step.

Session Key The encryption key used for a particular session. This key should change between sessions.

Shared Secret A value, usually a password, shared between two systems but otherwise private.

Bibliography

[1] Jeremy Allison. Security soup. Jan 2004. http://www.linux.org.au/conf/2004/eventrecord/LCA2004-cd/papers/securitysoup.sxi.

[2] Andrew Bartlett. Using samba as a pdc. *Linux Magazine*, Feb 2002. http://www.linux-mag.com/2002-02/samba_01.html.

[3] Andrew Bartlett. Beyond samba - intergrating windows authentication into a wireless vpn solution, May 2004. http://hawkerc.net/staff/abartlet/comp3700.

[4] J Brezak and M Swift. *The Microsoft Windows 2000 RC4-HMAC Kerberos encryption type*, May 2002. http://ftp.ist.utl.pt/pub/drafts/draft-brezak-win2k-krb-rc4-hmac-04.txt.

[5] John Brezak. *Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources*, 2002. http://msdn.microsoft.com/library/en-us/dnkerb/html/msdn_pac.asp.

[6] Bill Bryant. Designing an authentication system: a dialogue in four scenes. http://web.mit.edu/kerberos/www/dialogue.html. February 1988.

[7] Cato Institute. Encryption policy / export controls. http://www.cato.org/tech/encryptionandwiretapping.html.

[8] D. Chadwick. *Understanding X.500 - The Directory*. 1994. ISBN 185-0322-813. http://www.isi.salford.ac.uk/staff/dwc/X500.htm.

[9] Roger Clarke. Authentication: A sufficiently rich model to enable e-business. http://www.anu.edu.au/people/Roger.Clarke/EC/AuthModel.html. Dec 2001.

[10] Andrew Tridgell et al. Pidl, 2004. http://download.samba.org/ftp/unpacked/samba4/source/build/pidl/.

[11] Gerald Combs et al. Ethereal: A network protocol analyzer. http://www.ethereal.com/. 2004.

[12] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. Http authentication: Basic and digest access authentication. http://www.ietf.org/rfc/rfc2617.txt. Jun 1999.

[13] Free Software Foundation. Free software foundation, . http://www.fsf.org/.

[14] Free Software Foundation. Gnu general public licence, June 1991. http://www.gnu.org/copyleft/gpl.html.

[15] Ariel Futoransky and Emilano Kargieman. ssh insertion attack. http://www2.corest.com/common/showdoc.php?idx=131\&idxseccion=10. 1998.

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

[17] Gracion Software. What is LDAP? http://www.gracion.com/server/whatldap.html.

[18] Heimdal. Heimdal. http://www.pdc.kth.se/heimdal/.

[19] Chris Hertel. *Implementing CIFS*. Prentice-Hall, 2003. ISBN 0-13-047116-X. http://www.ubiqx.org/cifs.

[20] T. Howes. *The String Representation of LDAP Search Filters*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2254.txt. RFC 2254.

[21] T. Howes and M. Smith. *The LDAP URL Format*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2255.txt. RFC 2255.

[22] Tarjei Huse. *HeimdalKerberosSambaAndOpenLdap*, 2004. https://sec.miljovern.no/bin/view/Info/HeimdalKerberosSambaAndOpenLdap.

[23] Love Hörnquist Åstrand. Re: svn commit: lorikeet r43 - in trunk/heimdal/lib: hdb kadm5. http://lists.samba.org/archive/samba-technical/2004-September/037075.html. Sep 2004.

[24] isode. Ldap and x.500. *Messaging Magazine*, Sep 1996. http://www.isode.com/whitepapers/ic-6033.html.

[25] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*, September 1993. ftp://ftp.isi.edu/in-notes/rfc1510.txt. RFC 1510.

[26] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*, February 1997. ftp://ftp.isi.edu/in-notes/rfc2104.txt. RFC 2104.

[27] Paul Leach and Dan Perry. Cifs: A common internet file system. *Microsoft Interactive Developer magazine*, Nov 1996. http://www.microsoft.com/mind/1196/cifs.asp.

[28] Volker Lendecke. Advances in samba4. http://www.samba.org/samba/news/articles/samba4_vl.pdf. Aug 2004.

[29] Itsik Mantin. RC4. http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html.

[30] Jim McDonough. Implementing an active directory domain controller: Obstacles, hurdles, and the finish line. http://www.sambaxp.org/sambaXP_2004/06-mcdonough-ads.pdf. 2004.

[31] Microsoft Corporation. Windows server 2003 technical reference. http://www.microsoft.com/Resources/Documentation/windowsserv/2003/all/techref/en-us/w2k3tr_sec_authn_over.asp. .

[32] Microsoft Corporation. *The Security Support Provider Interface*, 1999. http://www.microsoft.com/windows2000/techinfo/howitworks/security/sspi2000.asp.

[33] Microsoft Corporation. *User Authentication with Windows NT*, 2001. http://support.microsoft.com/default.aspx?scid=kb;EN-US;q102716.

[34] Microsoft Corporation. *DsCrackNames*, May 2004. http://msdn.microsoft.com/library/en-us/ad/ad/dscracknames.asp.

[35] Microsoft Corporation. *How to enable NTLM 2 authentication*, 2004. http://support.microsoft.com/default.aspx?scid=kb;en-us;Q239869.

[36] Microsoft Corporation. Using the midl compiler, Oct 2004. http://msdn.microsoft.com/library/en-us/midl/midl/using_the_midl_compiler_2.asp.

[37] MIT. What is the export status of kerberos?, . http://www.faqs.org/faqs/kerberos-faq/general/section-15.html.

[38] MIT. Kerberos: The network authentication protocol. http://web.mit.edu/kerberos/www/. 2004.

[39] Mudge. L0phtcrack 1.5 lanman / nt password hash cracker. http://www.insecure.org/sploits/l0phtcrack.lanman.problems.html. 1997.

[40] Open Group. *The Open Group ActiveX Core Technology Reference*, chapter 11 - NTLM. . http://www.opengroup.org/comsource/techref2/NCH1222X.HTM.

[41] Open Group. *DCE 1.1: Remote Procedure Call*, 1997. http://www.opengroup.org/onlinepubs/9629399/toc.htm.

[42] R. Rivest. *The MD4 Message-Digest Algorithm*, April 1992. ftp://ftp.isi.edu/in-notes/rfc1320.txt. RFC 1320.

[43] Ronald L. Rivest and Alan T. Sherman. Randomized encryption techniques. http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C82/145.PDF. Also appears as Technical Memorandum TM-234, MIT Laboratory for Computer Science (January 1983), 1982.

[44] RSA Laboritories. What is RC4? http://www.rsasecurity.com/rsalabs/faq/3-6-3.html. 2003.

[45] Samba Team. *WHATS NEW IN Samba 3.0*, Sep 2003. http://www.samba.org/samba/history/samba-3.0.0.html.

[46] Samba Team. Samba team, 2004. http://www.samba.org/samba/team/.

[47] Burce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., second edition, 1996. ISBN 0-471-12845-7.

[48] Gursharan S. Sidhu, Richard F. Andrews, and Alan B. Oppenheimer. *Inside Appletalk*. Addison Wesley, 2nd edition edition, 1990. ISBN 0201550210. http://developer.apple.com/macos/opentransport/docs/dev/Inside_AppleTalk.pdf.

[49] SNIA CIFS Working group. *Common Internet File System Technical Reference*, 2002. http://www.snia.org/tech_activities/CIFS/.

[50] Sanj Surati and Michael Muckin. Dec 2002. http://msdn.microsoft.com/library/en-us/dnsecure/html/http-sso-2.asp.

[51] John Terpstra. *Samba 3.0 by Example*, chapter 8. Migrating NT4 Domain to Samba-3. Prentice Hall, 2004. http://www.samba.org/samba/docs/man/Samba-Guide/migration.html.

[52] Andrew Tridgell. Network analysis techniques. 2002. http://download.samba.org/samba/ftp/slides/net_analysis.pdf.

[53] Andrew Tridgell. Ten years of samba. http://www.samba.org/samba/docs/10years.html. Jan 2002.

[54] Andrew Tridgell. Andrew tridgell. http://samba.org/~tridge/. 2004.

[55] Ronald Tschalar. Ntlm authentication scheme for http, 2003. http://www.innovation.ch/java/ntlm.html.

[56] Jelmer R. Vernooij, John Terpstra, and Gerald (Jerry) Carter, editors. *Samba Official Samba-3 HOWTO and Reference Guide*, chapter 3. Server Types and Security Modes. Prentice Hall, 2003. http://us1.samba.org/samba/docs/man/Samba-HOWTO-Collection/ServerType.html.

[57] VMware. VMware GSX Server 3.1. http://www.vmware.com/products/server/gsx_features.html. 2004.

[58] M. Wahl. *A Summary of the X.500(96) User Schema for use with LDAPv3*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2256.txt. RFC 2256.

[59] M. Wahl, A. Coulbeck, T. Howes, and S. Kille. *Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2252.txt. RFC 2252.

[60] M. Wahl, T. Howes, and S. Kille. *Lightweight Directory Access Protocol (v3)*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2251.txt. RFC 2251.

[61] M. Wahl, S. Kille, and T. Howes. *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*, December 1997. ftp://ftp.isi.edu/in-notes/rfc2253.txt. RFC 2253.

[62] Assar Westerlund and Johan Danielsson. Heimdal and windows 2000 kerberos: How to get them to play together. pages 267–272, 2001. citeseer.ist.psu.edu/westerlund01heimdal.html.

[63] Joe Wilcox. Microsoft extends nt's life, 2003. http://news.com.com/Microsoft+extends+NT's+life/2100-1001_3-982220.html.

[64] Ross N. Williams. sep 1996. ftp://ftp.rocksoft.com/papers/crc_v3.txt.