Joining Security Realms:
A Single Login for Netware and Kerberos

William A. Adamson, Jim Rees, and Peter Honeyman
Center for Information Technology Integration
University of Michigan

Joining Security Realms: A Single Login for NetWare and Kerberos

William A. Adamson
andros@citi.umich.edu

Jim Rees
Jim.Rees@umich.edu

Peter Honeyman
honey@citi.umich.edu

Center for Information Technology Integration
University of Michigan

ABSTRACT

Accommodating disjoint security realms is a challenge for administrators who have to maintain duplicate data sets and for users who need to recall multiple pass phrases, yet joining security realms together can expose one realm to the weaknesses of the other. In this paper, we compare the Kerberos and NetWare security realms, examine methods of joining the two realms under a single login, and propose an attractive single login design.

1. Introduction

The Information Technology Division (ITD) of the University of Michigan provides a wide range of computing resources to its users, several of which include security services. To facilitate access control and accounting for the use of these resources, a single security service is needed that covers a wide range of the provided computing resources. Kerberos IV[1] is ITD's security service of choice and is deployed on all IFS[2] client machines. Yet, NetWare 4.0,[3] an increasingly popular choice in the University's information technology environment, uses its own security service, one that is incompatible with Kerberos.

In this paper, we examine the feasibility of a *single login* for the Kerberos and NetWare security realms. By single login, we mean that from any platform a user types one user name/pass phrase[4] pair to obtain access to all services in the computing environment. We approached this problem with the following design goals:

- A single client login program to give the user access to both Kerberos and NetWare services, using a single user name and pass phrase for DOS/Windows clients.
- No changes to Kerberos.
- Compatibility with existing NetWare applications and services.
- No reduction in security due to the single login, in either security realm.

The notion of single login holds implications from both the client and security server point of view. Since a user in both realms enters the same pass phrase, client security issues in both realms are merged. On the server side, the pass phrase data bases of the two realms are now related; this relationship determines the server side exposure incurred by each realm due to the single login. The goal of this paper is to understand the implications of these mergers, and to propose a satisfactory design.

The remainder of this paper is organized as follows. In the first section, we give a short overview of Kerberos and NetWare security services. The

1. Throughout the remainder of this paper, non-specific references to Kerberos refer to the version of Kerberos IV in AFS 3.*x*.
2. The Institutional File System, based on AFS, is deployed by ITD.
3. Throughout the remainder of this paper, references to NetWare refer to NetWare 4.0.
4. Because "password" often connotes a dictionary word, we choose to use the more general term, "pass phrase."

second section lists some known current security lapses in the two security realms, organized by attacks. The third section discusses security lapses that might be caused by single login. The fourth section presents two single login designs. In the last section, we describe our conclusion, and discuss ways to increase the level of security of the single login.

2. Overview of Security Services

We assume the reader is familiar with Kerberos and NetWare security realms, and offer the following overview.

2.1 Kerberos IV

Kerberos [1] is a trusted third-party authentication service based on the challenge-response model of Needham and Schroeder [2]. Each client trusts Kerberos' judgement as to the identity of each of its other clients. Kerberos keeps a database of its clients, which it calls principals, and their keys. The key is referred to as a *symmetric* key because the same key is used for both encryption and decryption of data. Because the `kaserver`[5] knows these keys, it can create messages that convince one principal that another is who it claims to be. The `kaserver` also generates temporary secret keys, called session keys, which can be used for authentication or privacy of two parties.

Login proceeds as follows. The user is prompted for a uniqname[6]. Once it has been entered, a request is sent to the `kaserver` containing the uniqname and the name of a special service known as the ticket-granting service. If the client is known to the `kaserver`, a random session key and a "ticket" for the ticket-granting server is returned. This ticket contains information such as the uniqname, the name of the ticket-granting server, the current time, a lifetime for the ticket, the client's IP address, and the random session key just created. This is all encrypted in a key known only to Kerberos and the ticket-granting server, making the contents of the ticket unknowable to others.

The `kaserver` then sends the ticket, a copy of the random session key, and some other information back to the client all encrypted in the user's key. Once this response has been received by the client, the user is prompted for a pass phrase. The pass phrase is converted into a DES key and used to decrypt the response. The ticket and the session key along with some other information are stored and used for background authentication[7]. The user's pass phrase and DES key are erased from memory. The cleartext pass phrase is never transmitted over the network.

Some Kerberos weaknesses are described by Bellovin and Merritt [3].

2.2 NetWare Authentication

NetWare Core Protocol (NCP) authentication uses RSA Data Security, Inc's MD-4 Message-Digest Algorithm [4]. This algorithm uses mutual authentication: each end of a connection verifies its identity to the opposite end, in contrast to the trusted third party authentication method of Kerberos. The RSA cryptosystem uses a public/private key pair where the public key is used to encrypt data, and the private key to decrypt data. Since keys are not generated from the users' pass phrase, the NetWare client must obtain the user's private key from NetWare Directory Service (NDS) at login.

Upon boot, the client agent responds either to the nearest NDS tree broadcast, or to a preferred NDS tree name stored in the local file system, and connects to the service [5]. This connection lasts until reboot. As the SITES[8] are currently configured, the login program is not stored locally in the client file system, but resides on the server. The client requests the login program at boot.

The login program sends the user's login name to the server, and receives the NDS server's public key and the user's NDS User ID (*figure 1*, steps 2-7). The server also sends a four byte pseudorandom number as a challenge.[9] Note that these NCP requests and responses are not authenticated with

5. In AFS, the Kerberos database is managed by a progra called `kaserver`.
6. Uniqname is a University of Michigan program that ensures that user-assigned logins are unique campus wide, regardless of platform.
7. The term background authentication refers to the process of authenticating to a service by using some cached credential derived from the user's pass phrase, obviating the need for the user to re-enter the pass phrase.
8. SITES is responsible for deployment of NetWare for ITD on the University of Michigan campus.
9. We assume throughout this paper that any pseudorandom number generator used by Kerberos or NetWare is strong.

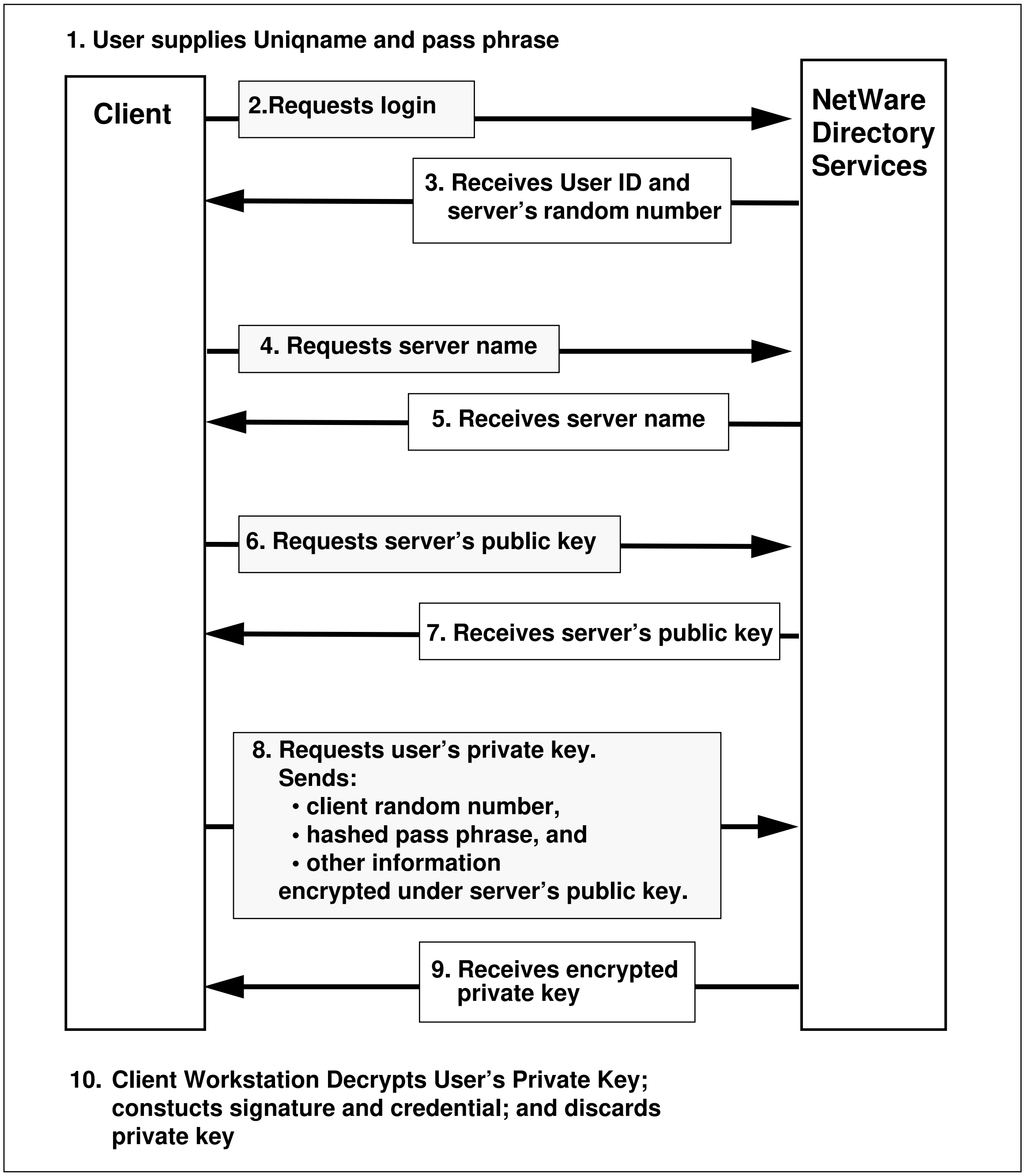

Figure 1. NetWare Authentication

MD-4 signatures, as the MD-4 state is not initialized until after a successful login.

The client login creates a hash of the pass phrase using the information it received from the server. The client agent then generates a random four byte challenge and encrypts the hashed pass phrase, the server challenge, and some other information with the server's public key and sends it all to the server [4] (*figure 1* step 8). The cleartext pass phrase is never transmitted over the network.

The server decrypts the packet using its private key. Using information derived from the pass phrase stored in NDS at pass phrase creation, the user's NDS ObjectID, and the random number challenge it sent to the client, the server repeats the hash calculation performed on the client and compares the result with the hashed pass phrase received from the client. If they prove to be equal, the server has a high degree of confidence that the agent submitting the hashed pass phrase is the same agent that generated the pass phrase derived information stored in NDS [6]. In other words, the client's ability to create the hashed pass phrase authenticates the connection.

The NDS server, assured that the user's identity is as claimed, retrieves the user's private key, which is stored in NDS. The server encrypts the private key along with the challenge received from the client, and sends it all to the client (*figure 1* step 9). The client agent decrypts the packet and compares the received challenge with the one it sent to the server. The client now has its private key and can use MD-4 signatures in its NCP exchanges.

The client uses the private key to encrypt a credential, which is a user and session identification data structure, to create a signature. Once the signature is created, the private key is erased from memory (*figure 1* step 10). The signature never gets transmitted over the network. Instead a "proof", derived from both the signature and message data, is constructed and transmitted with each request or message as the authenticator [7]. The signature/proof mechanism is used for background authentication.

NetWare does not provide a means for applications to use data encryption. Neither the RSA public/private key pairs nor the RSA encryption/decryption API are exposed.

3. Current Exposure Under Dual Login

This section enumerates the methods to obtain pass phrases or another user's cyberspace identity (i.e. keys) and examines the response of both security realms to these attacks.

3.1 User Responsibility

Any security system requires users to identify themselves to the system. Absent the use of physical traits (e.g. retina scans), a user needs to possess a secret to present to the system, typically a pass phrase. The user is responsible for guarding this secret. Users need to pick good pass phrases and to guard against "shoulder surfing" (allowing others to watch them type in their pass phrase). They should also avoid writing pass phrases on paper or storing them where others might find them.

One way to reduce the risk from shoulder surfing and paper pass phrases is the use of secure cards. These cards produce a stream of one-time pass phrases using a seed known only to the card and the security server. Since physical possession of the card is required to log in, theft is more difficult and more easily detected. There is currently no support for the use of secure cards in either Kerberos or NetWare.

3.2 Corrupt Administrators

A dishonest system or network administrator is in a good position to steal pass phrases. The system is only as trustworthy as the people who run it.

3.3 Trusted Servers

Both Kerberos and NetWare require that their server machines be kept physically secure. If an attacker gains access to server machines, then she could collect pass phrases, cause denial of service, or install new or replacement programs, all of which result in unpredictable and undesirable server behavior. Therefore server machines should be kept in a locked room with access limited to authorized staff.

3.4 Client Trojan Horse

Client machines are typically not physically secure, which presents the opportunity to install a program (a "Trojan horse") that steals user pass phrases. The most obvious target of a Trojan horse attack is the login program itself, but other targets should not be overlooked. On some systems, it is possible to install a "keyboard sniffer" that records individual keystrokes, unknown to users. Even seemingly benign application programs can be used in a Trojan horse attack, as they typically inherit all the user's permissions when they run. Naive users may be duped into supplying a pass phrase when a Trojan horse demands it, even if there is no obvious need for the program to have the pass phrase.

On UNIX clients, an attacker must boot the system in single user mode or otherwise obtain root permissions to access the local filesystem and install the Trojan horse. On the other hand, Mac and PC clients have no local filesystem access control, so a Trojan horse is trivial to install. IFS clients consist of all Transarc-supported UNIX clients, Macs, and soon DOS/Windows PCs. There are also unsupported copies of a cache manager for OS/2. NetWare clients consist of DOS and Windows PCs and Macs. Unless these machines are physically secure, all are equally open to client Trojan horse attacks.

Clients can be configured to automatically "scrub," i.e. re-install much of their software, after each user logs out. This is time-consuming but reduces the risk that a Trojan horse planted on a client remains when the next user logs in.

3.5 Promiscuous Access

There are products such as Network General's Sniffer and operating systems such as AIX for the IBM RS/6000 that allow promiscuous access to network traffic. In a large and diverse computing environment where it is impossible to secure the network completely, we must assume that any traffic may be sniffed. Sniffing provides the data for spoofing, replay, and dictionary attacks. Sniffing also allows for capturing any pass phrases that are passed on the network in the clear.

Several heavily used protocols place pass phrases on the wire in cleartext. FTP conveniently places the whole pass phrase in one packet. TELNET generally places each character of the pass phrase in a separate packet. RLOGIN and some versions of POP also place the pass phrase on the wire in the clear. Any Kerberos or NetWare user that uses these protocols exposes her pass phrase to sniffing.

Pass phrases may be particularly vulnerable to sniffing at the time they are set or changed, as a new secret (pass phrase or user key) must be generated by the client and sent to the server. In Kerberos, the user key is sent encrypted by a session key. In NetWare, the hashed pass phrase is sent encrypted by the user's public key.

3.6 Replay or Forged Packet

Replay attack consists of inserting a modified old packet into a privileged session in order to grant rights to the intruder. A new forged packet may also be used for this purpose. Kerberos makes use of timestamps and nonces, and NCP uses sequence numbers and the MD-4 message digest to minimize replay and forged packet attacks.

3.7 Dictionary Attack

Dictionary attack consists of passing every word in a dictionary as a possible pass phrase to a login or key generating program, and verifying the results either by attempted login or by comparing to the generated key. The dictionary can contain the entire English language, foreign languages, custom entries composed of past pass phrases, user's favorite places, slang, etc. NetWare limits the number of login attempts to hamper on-line dictionary attack.

Kerberos IV is subject to an off-line dictionary attack of the TGT [3]. The attacker asks for a TGT, claiming to be the victim. Kerberos returns a TGT encrypted in the victim's key. Because some of the contents of a TGT are well known (such as realm, and uniqname), and the string-to-key functions are public, a dictionary attack on the TGT can be launched; furthermore, because the attacker is in possession of the victim's TGT, the attack can be accomplished off-line. If successful, the attacker discovers with the victim's key, which can be used to assume the victim's Kerberos identity.

This method works not only because the form of the TGT is known, but also because the victim's key is formed from a pass phrase. The more cryptic the pass phrase, the less likely it is to be found in a dictionary. Kerberos session keys are derived from pseudorandom numbers, so they are (presumably) immune to dictionary attack. Kerberos V prevents this attack through the use of a pre-authentication step.

NetWare's RSA public/private keys are generated from pseudorandom numbers and are stored under the protection of a pass phrase. Pseudorandom keys are (presumably) much stronger than Kerberos user keys and so are immune to dictionary attack.

There is a sophisticated off-line dictionary attack on the NetWare protocol that spoofs the NDS server to obtain the user's hashed pass phrase. The attacker uses a client and the spoofed NDS server to obtain hashed guessed pass phrases, which are compared to the user's hashed pass phrase.

In this scenario, the attacker constructs a spoofing NDS server that responds to the client's initial connection exchange (*figure 1* steps 2, 4, and 6), masquerading as a legitimate NDS server. When a client makes an NDS request, the spoofer generates a public/private key pair and responds to the client with a User ID (a 32-bit number), a random challenge, and the generated public key (*figure 1* steps 3, 5, and 7). The client then sends the hash of the user's pass phrase encrypted under this public key (*figure 1* step 8). The fake server obtains the hash of the user's pass phrase by decrypting the packet with the private key.

Now an off-line dictionary attack proceeds as follows. The attacker uses a client to talk to the spoof server off-line, feeding it possible pass phrases. The spoof server responds with the same User ID and random challenge that it provided to the client, gathers the generated hash of the guessed pass phrase, and compares it to the hash of the user's pass phrase, collected from the client above. A match means that the user's pass phrase has been discovered.

This attack requires detailed knowledge of NCP and SAP (NetWare's advertising protocol) as well as an RSA public/private key engine.

3.8 Spoofing

Spoofing is pretending to be a server or a peer. Kerberos is immune to spoofing attacks as long as the servers are physically secure and their `/etc/srvtab` files are unavailable. The NetWare NDS server can be spoofed. We have identified two attacks that use an NDS spoof server.

SITES runs its public NetWare clients with guest login only; client login is kept on the NDS server. At boot, NetWare clients connect to the nearest advertised (by broadcast) NDS tree. If the client login is not local, the first thing the client does is to import the login program from the NDS server. An NDS spoof server can be on the local LAN. If its load is minimal, it can easily be the first to respond to a client's connection request and provide a Trojan horse login program to gather pass phrases. This provides the attacker with a distributed client Trojan horse.

If the client login is local, the intruder's NDS can respond to the client's login program's requests (User ID, server public key, random number, etc.) and receive the client's hashed pass phrase. The hashed pass phrase can then be dictionary attacked off line, as described above. This spoof/dictionary attack requires intimate knowledge of NetWare's NCP and SAP protocols, as well as the ability to generate RSA keys and use them for encryption/decryption.

Both these spoofs depend on the ability to spoof the initial client connection to the desired NDS tree.

3.9 Summary of Exposures

Both security realms contain lapses that allow an attacker to obtain pass phrases. Both realms share the protocol sniffing and local client Trojan horse security lapses, and each realm is open to a dictionary attack.

Kerberos exposures include:

- Sniff FTP, TELNET, or RLOGIN protocols to obtain pass phrases.
- Install client Trojan horse.
- Ask for a TGT and dictionary attack it off-line.

NetWare security exposures include:

- Sniff FTP, TELNET, or RLOGIN protocols to obtain pass phrases.
- Install client Trojan horse.
- Spoof as NDS and distribute Trojan horse to clients.
- Spoof as NDS and gather hashed pass phrases; dictionary attack off-line to obtain pass phrase.

4. Additional Exposure Due To Single Login

Creating a single login creates additional client-side and server-side exposure issues as described below.

4.1 Client-Side Issues

Kerberos and NetWare security realms are equally open to client Trojan horse attacks. Combining the realms exposes Kerberos pass phrases to the NetWare distributed client Trojan horse spoof, and so hampers the current Kerberos security. The distributed client Trojan horse exposure can be removed by providing local login programs to all SITES NetWare clients. An attacker can still install Trojan login programs, but must visit each client individually to do so.

4.2 Server-Side Issues: Pass Phrase Data Base Relationships

At ITD, we use Kerberos to authenticate to many services, so we view the Kerberos pass phrase data base as the master database and NetWare NDS as the slave. Thus, the Kerberos data base will not change; the NetWare data base will. Given this decision, the design question becomes what NetWare pass phrase will be presented to NDS and how it is related to the Kerberos pass phrase. Note that we are not talking about what pass phrase the user presents to the client; that will be the Kerberos pass phrase. Rather, we are talking about what goes on behind the scenes. There are three possibilities:

- NetWare pass phrase is the *same* as the Kerberos pass phrase
- NetWare pass phrase is *derived* from the Kerberos pass phrase.
- NetWare pass phrase is *unrelated* to the Kerberos pass phrase.

In the first case, where the same pass phrase is used in each realm, each realm's security is now dependent upon the other. The ability to compromise either system and obtain a pass phrase means that both systems are compromised. In the next section, we describe a candidate architecture that uses this scheme.

Carnegie Mellon University's NetWare AFS Project [8] proposes a single login solution of the second variety: the NetWare pass phrase is derived from the Kerberos client's key. Security in the NetWare realm depends on Kerberos but Kerberos security is only partially dependent upon the NetWare realm: compromising NetWare and obtaining the NetWare pass phrase exposes the Kerberos client key, which is less of a security threat than losing the Kerberos pass phrase.

In the third case, the NetWare pass phrase is unrelated to the Kerberos pass phrase. In the next section, we describe a candidate architecture that maintains a data base of NetWare pass phrases encrypted under user Kerberos keys. In that design, the NetWare pass phrase gives no access to the Kerberos security realm, while the Kerberos pass phrase gives complete access to the NetWare security realm.

Another potential scheme would be to generate a new random NetWare pass phrase for each login, forcing a pass phrase change in NDS. In this scheme, obtaining a pass phrase in one realm gives no access to the other realm, but this seems to be quite clumsy. Since the user does not know her NetWare pass phrase, unmodified NetWare client login will not work.

5. Single Login Designs

We describe two single login designs, noting their respective advantages and disadvantages.

5.1 Common Pass Phrase

In this design, the same pass phrase is used for both Kerberos and NetWare security realms. Users are allowed to change their Kerberos pass phrases freely, while ACL's on the User Object in NDS prevent them from changing their NetWare pass phrases directly. The NDS pass phrase is synchronized with the Kerberos pass phrase at the next client login as described below. The Kerberos and NetWare pass phrase data bases are then kept in synchrony until a pass phrase change.

There are two components to this design, `S1_LOGIN.EXE`, a new NetWare PC client login program, and `NW_AUTH.NLM`. This NLM[10] runs on NDS servers, and is a Kerberos service provider with NDS administration privileges that performs the pass phrase synchronization.

`S1_LOGIN.EXE` chains together normal Kerberos and NetWare login, gathering Kerberos tickets and NetWare credentials. After obtaining the uniqname and pass phrase from the user, Kerberos tickets are obtained in the usual way, and NetWare login is attempted (*figure 2*). If the data bases are synchronized, this succeeds and `S1_LOGIN.EXE` returns.

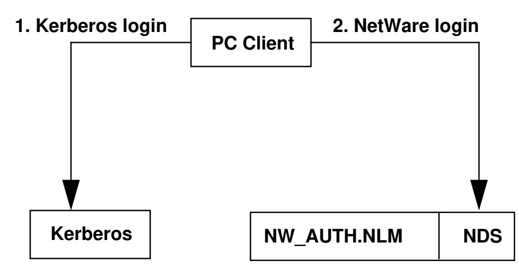

Figure 2. Common Pass Phrase Login: NDS is synchronized with Kerberos

10. NLM stands for NetWare Loadable Module.

If NetWare login fails, S1_LOGIN.EXE uses Kerberos to mutually authenticate with NW_AUTH.NLM. This provides a session key and a secure connection between the client and NDS. The uniqname and pass phrase are encrypted with the session key and sent to NW_AUTH.NLM, which forces a pass phrase change. The NetWare and Kerberos data bases are now synchronized, and normal NetWare login is tried again (*figure 3*).

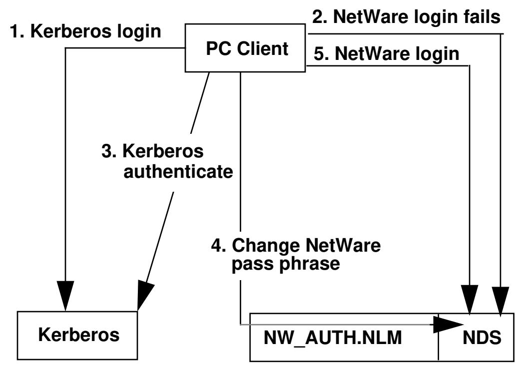

Figure 3. Common Pass Phrase: NDS change pass phrase. Kerberos pass phrase has been changed, NDS needs to be synchronized.

There are several advantages to this design. S1_LOGIN.EXE uses normal NetWare login, so a NetWare client login works "out of the box." If the user changes her Kerberos pass phrase on an unmodified Kerberos client, she does not run S1_LOGIN.EXE, and unmodified NetWare clients will continue to work with her old Kerberos pass phrase. Migration from NetWare-only to Kerberos+NetWare clients is accomplished by installing the modified client software, obviating a "flag day." The design also scales well, as communication with NW_AUTH.NLM occurs only upon a pass phrase change.

The major drawback to this design is that NetWare and Kerberos security realms are each forced to depend on the other's ability to protect the mutual pass phrase. Beyond that, a pragmatic concern arises due to the time-consuming nature of NDS synchroniztion. In the unmodified NetWare API, pass phrase change entails unsealing the public/private key pair stored in NDS under a key derived from the old pass phrase, and resealing them under a key derived from the new pass phrase. In our design, the old pass phrase is not available at NDS synchronization time, so the user's public/private key pair are no longer usable. Consequently, a new key pair must be generated, which takes significantly longer than does the unmodified NetWare login.

5.2 Distinct Pass Phrase

In this design, the NetWare pass phrase and the Kerberos pass phrase are unrelated, and there is no synchronization between Kerberos and NetWare pass phrase data bases. The user chooses Kerberos and NetWare pass phrases at account creation time. The Kerberos pass phrase is all that is needed to perform single login. The chosen NetWare pass phrase is stored in a NetWare pass phrase data base and retrieved by the single login program to obtain NetWare credentials.

Since NDS does not export pass phrases, it cannot be used as the NetWare pass phrase data base. This necessitates a modified NetWare change pass phrase program that synchronizes NDS with the NetWare pass phrase data base. As in the common pass phrase design, a user is not allowed to change her NetWare pass phrase directly. Instead, the modified pass phrase change program communicates with the NetWare pass phrase data base NLM, which performs the NDS pass phrase change.

There are three components to this design (figure 4):

- S2_LOGIN.EXE, a new NetWare PC client login program,
- NETPASSD.NLM, a daemon that maintains the NetWare pass phrase data base, and
- NEWPASS.EXE, a modified NetWare change pass phrase program

NETPASSD.NLM, which runs on NDS servers, is a Kerberos service provider with NDS administration privileges. It maintains a data base of NetWare pass phrases indexed by uniqnames. A global Kerberos DES key known only to NETPASSD.NLM is used to encrypt all stored NetWare pass phrases.

S2_LOGIN.EXE chains together normal Kerberos and NetWare login, gathering Kerberos tickets and NetWare credentials. After obtaining the uniqname and Kerberos pass phrase from the user, Kerberos tickets are obtained in the usual way. Kerberos is then used to mutually authenticate with NETPASSD.NLM, providing a session key and secure connection, which is used to retrieve the NetWare pass phrase for the given uniqname from the NetWare pass phrase data base. The NetWare pass phrase is then used in a normal NetWare login (*figure 4*).

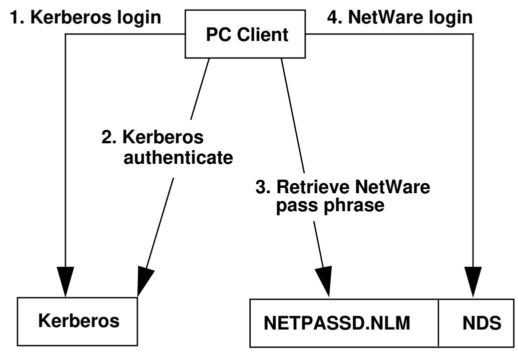

Figure 4. Distinct Pass Phrase Login

NEWPASS.EXE is called by the user to change the NetWare pass phrase. The user is required to have run S2_LOGIN.EXE before running NEWPASS.EXE. The user is prompted for her uniqname, old NetWare pass phrase, and new NetWare pass phrase. NEWPASS.EXE mutually authenticates with NETPASSD.NLM and establishes a secure connection for sending the uniqname, old, and new NetWare pass phrases. NETPASSD.NLM updates its data base and calls the normal NetWare change pass phrase API for NDS synchronization (*figure 5*).

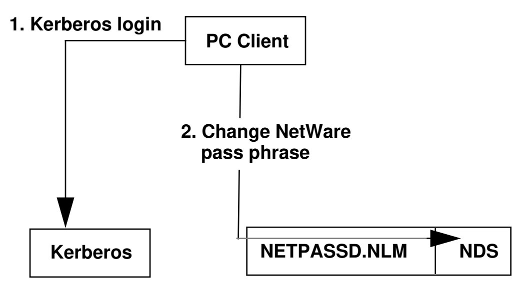

Figure 5. Distinct Pass Phrase: NDS change pass phrase. User must be logged in via Distinct Pass Phrase login.

The major advantage to this design is that given different NetWare and Kerberos pass phrases, compromising the NetWare security realm does not compromise the Kerberos security realm. Other advantages include an easy migration path from NetWare-only to Kerberos+NetWare clients. Since unmodified NetWare clients will work, modified client software can be installed as needed.

There are several disadvantages to this design. Since NDS doesn't export pass phrases, yet another data base needs to be implemented, kept synchronized, and made available, usually implying replication. Communication with NETPASSD.NLM occurs at each client login generating network traffic and promoting scaling as an issue.

Users are also required to choose and remember two pass phrases. Many users will choose to use the same pass phrase for both realms, making moot the first advantage cited above.

6. Conclusion

We conclude that a merging the Kerberos and NetWare security realms is feasible without altering the security of either. Pass phrase sniffing of FTP, TELNET, etc. protocols remains a concern, as does the client Trojan horse attack common to both security realms. We suggest that the NetWare client login program be a local executable, removing the threat of the distributed client Trojan horse spoof.

Dictionary attack of the Kerberos IV TGT and the NetWare 4.0 hashed pass phrase remain a problem. In security realms that we administer, we regularly attempt dictionary attack on all accounts and disable accounts whose pass phrase is thus revealed.

We feel that data put on the wire encrypted by an NDS public or private key is at least as safe as data put on the wire encrypted with the Kerberos session key [9]. Both of these keys are seeded with a random number, immunizing them from dictionary attack.

We described two single login designs that meet the remaining goals of giving user access to both Kerberos and NetWare services with no changes to Kerberos and compatibility with existing NetWare applications and services.

The common pass phrase design is simple, scales well, and seems to be easy to manage, and is being deployed by ITD. We can raise the security level of this design in several ways. Encrypting the pass phrase sent to NW_AUTH.NLM for NDS synchronization with an RSA public key instead of (or in addition to) the Kerberos session key might increase the protection of the pass phrase. This would require NetWare to expose RSA encryption for data transfer. Authenticating the initial client to NDS connection using Kerberos would prevent NetWare server spoofing.

The distinct pass phrase design is similar to the single login design proposed by the DCE Security SIG based on work done by Chii-Ren of Citicorp [10]. The slave security realm's (e.g. NetWare) pass phrase is stored under the master security realm's (e.g. Kerberos) protection and retrieved automatically at login. This design is extensible to any number of slave security realms, and has the additional advantage of allowing for all pass phrases for all security services for a single user to be different. ITD is investigating using such a scheme as part of a complete data base architecture redesign effort.

ITD can increase the security level of its Kerberos service in several ways. Switching from Kerberos IV to Kerberos V would address the exposure to off-line dictionary attack [3]. Deploying Kerberized TELNET and FTP services would help eliminate cleartext pass phrases over the network.

7. References

1. J. G. Steiner, B.C. Neuman, and J.I. Schiller, "Kerberos, An Authentication Service for Open Network Systems," in *Proceedings of the Winter USENIX Conference*, Dallas (January 1988).
2. R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers." *CACM* **21**(12), pp. 933–999, (December 1978).
3. S.M. Bellovin and M. Merritt, "Limitations of the Kerberos Protocol." in *Proceedings of the Winter USENIX Conference*, pp. 253–267, Dallas (January 1991).
4. T. Myers, "RSA and Kerberos Technology Overview," Novell Technical Development Conference, (March 1992).
5. Novell Inc., *NetWare Programmer's Guide for C*, Version 1.0, pp. 249–276, (June 1993).
6. Novell Inc., "Encrypted Login for NetWare Service Requesters," p. 11, (July 1993).
7. Novell Inc., *Using NetWare Services for NLMs*, Edition 1.1, Chapter 9, "Authentication Operations" pp. 83–94, (1993).
8. J. Stein et al., *NetWare AFS Project*, Carnegie Mellon University Computing Services Special Projects and School of Computer Science, (April 13, 1994).
9. B. Schneier, *Applied Cryptography*, pp. 259–260, John Wiley & Sons, (1994).
10. C. Tsai, "Single-Logon Issues for Heterogeneous Distributed Computing," Citicorp International Communications, Inc., (October 1994).