Administering Windows NT Domains Using a non-Windows NT PDC

Gerald Carter
Engineering Network Services
Auburn University
jerry@eng.auburn.edu

Abstract

The chronicles kept by an explorer as he or she journeys into new territories can be invaluable for those who come later. The writing describe successes and failures, warning and hints to other possible solutions yet unfound. The evolution of a network is not different than the Lewis and Clark expedition of the early 1800's. Each day we learn something new. Either our proposed solution worked or it did not. It was either a dead-end or perhaps simply a detour.

In the Fall of 1997 with the release of Samba version 1.9.18alpha1, the College of Engineering at Auburn University began to experiment with the possibility of using a non-Windows NT machine (i.e. Samba running on a Solaris server) as a Primary Domain Controller for Windows NT desktop clients. This journey has continued through the present and is still progressing.

This paper will be a post-narrative of that journey. It is our belief that the administrative models and tools developed to support this environment will also be beneficial in other integrated environments, either in parts or as a whole.

This journey will be divided into in five topics: (1) relevant details regarding the configuration of the Samba PDC, (2) user account management, (3) remote file and printer access, (4) remote administration of clients and servers, and (5) the remote updates of systems, applications and associated settings. Each section will remain distinct enough so as to allow the reader to implement the varying portions of this paper independently.

Defining the Issue at Hand

The College of Engineering network is like many networks. We support a wide range of users each desiring different services. Some users require access to UNIX servers running various applications while others find one of the various Windows platforms to be their OS of choice.

When integrating heterogeneous networks, one possible solution is to maintain separate networks in parallel. For example, each network maintains the servers necessary for implementing services to support its associated clients even though each network may logically exist on the same wire. Services such as access to network printers are duplicated across server platforms. The parallelism is made transparent to the user by synchronizing the user account information.

The other possibility for integration is to centralize primary services such as user authentication on one server OS. The server must then support the functionality necessary to allow clients of other operating systems to become citizens of the network.

The College of Engineering at Auburn University has selected the second option. Our goal has been to provide continuity of services wherever possible without sacrificing the stability of these services. For example, users are able to access the same disk space as their home directory when logging into a PC or logging into a UNIX workstation, and network printers can also be accessed from either client platform. Perhaps the most important point in achieving this consistency is that each user is always authenticated with the same username / password pair.

Justifying the Trip

Regardless of where one is employed, any project that involves the expenditure of effort or resources requires some justification prior to approval. Therefore before continuing, I will provide the rational that began the exploration of using a non-Windows NT Domain Controller.

The first reason for entertaining this idea was the current existence of a stable Solaris based UNIX infrastructure. Global user authentication, as well as ubitquous remote file access, were already implemented. If possible, we wished to leverage off existing services, both hardware and software, rather than replicate and reimplement them.

Another reason for the desire to keep primary services on one server platform was the shortage of support staff to maintain another operating system. The addition of another OS in the server room does not increase support costs linearly but rather exponentially. The reason for this can be accounted for in keeping up to date with platform patches, system tuning, and general knowledge of the workings of an operating system.

Implementation of a Samba Controlled Domain

Many of the services provided by the Engineering network are based upon open source software projects. For example, BIND and Sendmail are used for DNS and mail service respectively. Therefore, we were comfortable with the idea of providing some of the support for server packages ourselves. With this in mind, Samba (www.samba.org) was selected as the CIFS server over other packages such as the TotalNET Advanced Server.

Details of the implementation of the Windows NT Domain Control protocol can be found in [**1**] as well as in documentation included with the Samba source code distribution. Currently Samba does not "officially" support Domain Control for Windows NT clients.

Of course this is not the same as domain control for Windows 9x and older Windows clients. These clients use an entirely different mechanism for user validation and are not members of a domain in the real sense of the word.

The reason for the unofficial label is that not all of the functionality has been implemented. For a current list of implemented features and progress on remaining functionality, the reader is referred to [**2**].

At the time of the writing, one of the major items which has not been completed is the capability to participate as a domain controller in inter-domain trust relationships. For this reason, the College of Engineering chose to run a single, global domain for all Windows NT clients located within the College of Engineering. Currently the number of NT clients is around 50 with the majority of these being located in a public workstation lab. The number of users supported in this domain is approximately 1,700 with over half being regular users of NT clients.

It would have been just as easy to configure multiple, isolated domains. The reason for this that replicating Samba's account database is trivial when done via a secured `rdist` or more preferably by using `scp`. Future work in Samba will allow for other database backends, such as LDAP which provide for even easier replication, to be used.

Because this paper is more concerned with the model used in administering a non-Windows NT Server controlled domain, the reader is referred to [**3**] and [**4**] for details regarding configuring Samba in this capacity.

There is however, one detail that is pertinent to later discussions contained within this paper. This is the method used for storing user accounts within Samba. I have already alluded to this when discussing replicating Samba's list of accounts.

The SMB (a.k.a. *C*ommon *I*nternet *F*ile *S*ystem) protocol supports two levels of transmission of user credentials, or passwords (see [**5**]). The first is to transmit the password in plain text. In this mode, Samba is able to validate user connections against the standard UNIX password database (i.e. /etc/password or the network equivalent such as a NIS map).

The second method of validating user credentials is to use a challenge-response exchange. The details of this are explained in [**6**]. This mode requires the SMB server have previous knowledge of the user's hashed password. When SMB password encryption support is enabled in Samba, the account password hashes are stored in a file generally named `smbpasswd`. This is not to be confused with the tool used to manually manipulate these entries that is also named `smbpasswd`.

In order to support domain logons from an NT client, Samba must be configured to support password encryption. It also should be mentioned that supporting encrypted passwords is an "all or none" situation on a per connection basis. The reason for this is that the encryption capabilities of the server are negotiated during the session setup when in user level security (the reader is referred to [**5**] for a description of connection setups in the CIFS protocol).

Therefore, maintaining a Samba PDC also requires maintaining a separate list of NT accounts which is separate from the accounts listed in `/etc/passwd`. In fact, the only joining field between the two account lists is the UNIX uid which must be valid to control access to the underlying file system.

User Account Management

This brings up the topic of user management in a Samba controlled Windows NT domain and how to synchronize this information with the existing UNIX user accounts. The first issue to be addressed was how to initially populate Samba's `smbpasswd` file given that the College currently supported over 6,000 active user accounts. An active account is defined as one that has been accessed in the past six months.

In order to generate the necessary hashes, Samba will need access to the plain text of a user's password. In the general case, this is impossible and at the very best impractical due to UNIX's non-reversible password encryption algorithm.

There are two plausible means of performing this account population. The first is to use Samba's `update encrypted` parameter and require that users connect to a Samba server that does not have password encryption enabled. Once the `smbd` daemon successfully validates the user's plain text password, it will generate the respective password hashes and store them in the `smbpasswd` file. Once all passwords have been captured, a site can successfully migrate to encrypted passwords transparent to the users.

The second solution is to develop a custom application that will capture the user's plain text password and then generate the hashes itself. Of course, the most obvious chance to capture the plain text of a password is when the user is changing his or her password. This is the solution explored in this paper. **Figure 1** illustrates what occurs when a user changes their password from an Engineering UNIX workstation.

Note that the password is only validated against the UNIX account entry (i.e. NIS). If the change is successful, then the new password is written to the `smbpasswd` file with no questions asked.

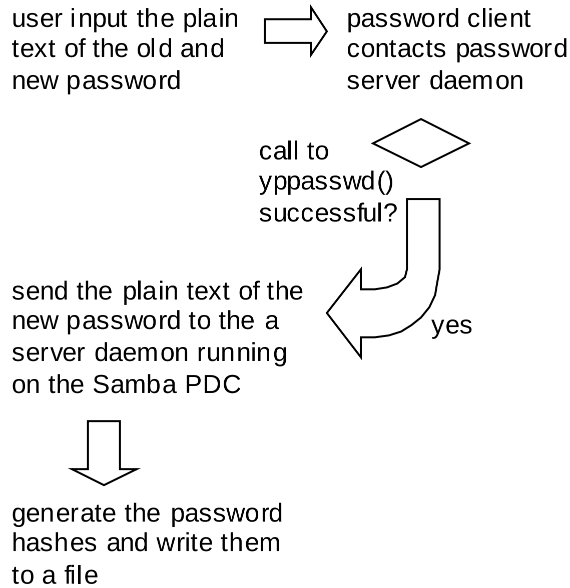

Figure 1: Flowchart of a user password change

By replacing the standard `/bin/passwd` with a custom in-house script, user accounts in the smbpasswd password file can be created using the same means for new and already existing accounts. This also provides a means to keep the password entry in both `/etc/passwd` and `smpasswd` synchronized.

For details on the availability of this tool, the reader is referred to [**10**]. Currently, the password change client is only available from UNIX hosts. However, plans are to include a Win32 binary that would allow Windows users to change their password directly from their desktop. This customized solution is more acceptable for us than attempting to support Windows' native method of changing passwords.

Remote File and Printer Access

For several years, PC's on the Engineering network gained access to remote file systems and printers via a locally installed NFS client. This was an acceptable solution at the time. However, there were several associated drawbacks.

The most prevalent one was the fact that NFS connectivity had no native support in DOS / Windows clients and therefore required an installation separate from the operating system itself. This proved to be problematic as systems were upgraded and incompatibilities arose between the newer operating systems and the older NFS client software.

These circumstances led us to search for solutions that would allow for utilizing the native networking support within the operating system itself. The search resulted in a decision to support a CIFS-enabled server. Windows clients would then be able to use built-in support for the Microsoft Networking model.

To implement a global domain for Windows 9x and Windows NT clients, multiple Samba servers running the latest stable release were installed to sit on top of existing NFS and LPR servers. The Samba PDC performed validation only. This allowed for us leverage off the latest domain control functionality with a minimal sacrifice of stability for file and print services. The `smbpasswd` file is then distributed to the servers using a secure method.

As can be seen in **Figure 2**, this allows for providing some of the consistency in logon environments mentioned previously. Hardware already was in place to support various disk configurations for home directories and network applications. Therefore, Samba servers initially acted as a gateway to these file systems. Over time, the Samba servers themselves are beginning to accumulate local disk space which is used for providing applications that only need be accessible from a PC such as the case with roaming user profiles.

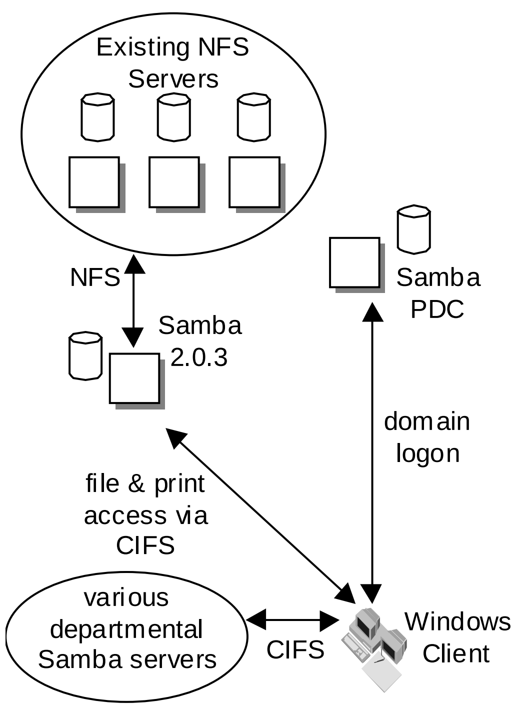

Figure 2: Overview of the College of Engineering network topology for PC's

The current system for our main Samba file server is a Sparc Ultra 170 with 384 Mb RAM running Solaris 2.6. The file server daily supports 400 - 500 Windows clients. Of these clients, normal use involves approximately 225 - 250 concurrent connections with each user having anywhere from 2 - 5 shares mounted. Statistics indicate that the load could comfortably increase to 300 connections with minimal detrimental effects given the current hardware configuration.

Network printers are accessed through the SMB server which then sends the spooled job to the remote printer via `lpr`. Rather than use the Solaris `lpsched` printing system, all servers are configured to use the Solaris port of the Berkeley `lpd` printing system.

Previously I mentioned that the Engineering Windows NT domain contained approximately 50 clients. This is true as this number reflects the actual Windows NT clients. However all PC's on the Engineering network utilize the file and printer services provided by various Samba servers. Domain control for Windows 9x and older Windows clients is provided by our main Samba box, which is currently running Samba 2.0.3.

Details of configuring Samba file and print servers can be found in the various documentation included with the Samba source code as well as various links from the main Samba web site (see http://samba.org).

Remote Administration of Clients and Servers

When administering a network, is it imperative to be able to automate common tasks such as process control, system reboots, and managing disk space. Maintaining a Samba controlled domain is, in many ways, identical to managing a true Windows NT domain. However, while some tools such as the Server Manager and User Manager for Domains allow access to information on the Samba PDC, these applications are very inefficient for managing a large number of domain clients. When attempting to perform these duties on a mass scale, network administrators can turn to scriptable solutions such as Perl, Python, or one of the Windows based programming languages. The advantage of using a cross platform scripting solution is the ability to leverage off current programming knowledge and the reuse of existing scripts.

There are many possibilities for remotely accessing NT clients from another NT box. However, remotely administering NT clients from a UNIX terminal window takes a little bit of creativity. Through the use of free RSHD service for Windows NT (home.us.net/~silviu), UNIX network managers are able to perform basic NT administrative tasks without leaving their desktop.

The service does have some caveats that must be mentioned. The first is that the service provides no user authentication whatsoever. Therefore all commands executed by the rshd daemon will be done within the context of the account under which it is currently running. By default this would the LocalSystem account. However, we have reconfigured the service to run as a local administrative account.

It is a requirement that it can be guaranteed, and with a great degree of certainty, who can possess the ability to `rsh` into a NT client. This is done by creating `%SystemRoot%\rhosts` and editing it to contain a single entry referring to the network NIS+ master. The file is then locked down with the appropriate file permissions to disallow viewing or modification by any account other than a local administrator. **Figure 3** illustrates access control to these machines.

It is the belief that if anyone was able to gain unauthorized access to our NIS+ master, the NT clients will become small potatoes. While not entirely correct, it is simply another example of leveraging off the existing UNIX infrastructure, in this case security. Also we have modified the `rshd` source to perform forward and reverse DNS lookups so as to ensure that the machine from which the `rsh` command is arriving from is actually who it says it is.

With the addition of the capability to execute commands remotely, the quest then becomes to build up a sufficient toolbox of command line tools to be able to perform necessary tasks. Many possibilities have been outlined in [**7**].

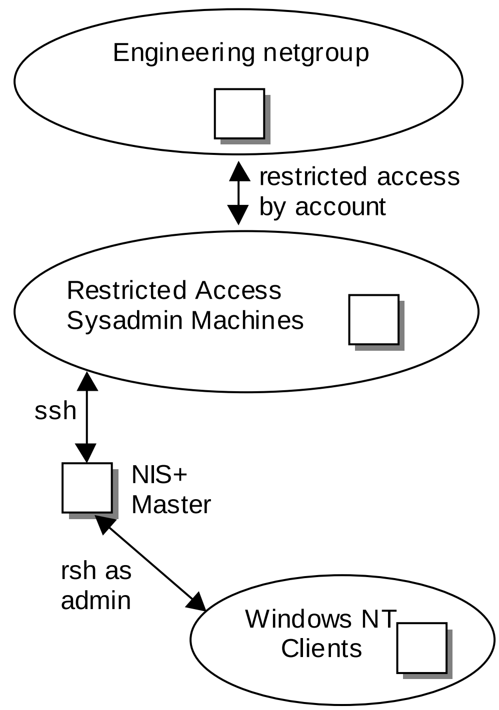

Figure 3: Restricting rsh access to Windows NT clients.

Here is an example of granting a Samba domain user named *toby* full permission to a share named *share1* on a Windows NT 4.0 domain member named *peeps*. The `sharegrant` utility is part of Pedestal Software's NT Security tools (www.pedestalsoftware.com).

```
$ rsh peeps 'sharegrant share1 toby:full'
Granting permissions to: ENG-NT\toby
```

Although the share ACL could be changed from another Windows NT machine, it cannot be guaranteed that one is always nearby. It is possible to always guarantee that telnet access to a UNIX workstation is available.

Perl has often been described as the kitchen sink of scripting languages and the Win32 port of the language is no exception. ActiveState's Win32 version of Perl5 (www.ActiveState.com) provides modules for managing NT user accounts, querying and modifying registry values, retrieving information from the NT EventLog, as well as the traditional capabilities such as sockets and regular expressions.

The combination of an RSHD service and Perl provides a means for doing just about anything one could wish. To illustrate this point, I have made available several scripts that perform such things as configuring a machine in a public computing lab by setting common registry values. Again the reader is referred to [**10**] regarding the availability of these scripts as well as updates. **Listing 1** contains the output from a perl script that queries the local EventLog and displays information similar to the UNIX `last` command. Remember that all of this information is gained from a UNIX terminal window.

Listing 1: Using a perl script to determine the list of recently logged on users on a Windows NT workstation.

```
$ rsh beeps 'perl last.pl' | head –10
username machine domain   logon - logoff
gaoyun1  DRACO   ENG-NT   May 31 10:46 1999 - still logged on
jingcai  DRACO   ENG-NT   May 30 17:51 - May 31 00:30 1999
yizhou   DRACO   ENG-NT   May 30 01:13  - May 30 01:18 1999
jingcai  DRACO   ENG-NT   May 29 18:56  - May 29 23:01 1999
millean  DRACO   ENG-NT   May 29 18:13  - May 29 18:40 1999
```

If we then wanted to find out which processes were currently running on beeps, we could use the `pulist.exe` command included with the Windows NT 4.0 Server Resource Kit [**8**]. Note that some of the output has been deleted for brevity.

Listing 2: Retrieving the list of running process from a Windows NT machine using the pulists.exe command.

```
$ rsh beeps '\\ivy\bin\ntreskit\pulist'

Process        PID     User
Idle           0
System         2
 ……
rshd.exe       124     BEEPS\admin
NDDEAGNT.EXE   157
EXPLORER.EXE   130
systray.exe    59
F-AGENT.exe    191
OSA.EXE        188
xwin32.exe     190
CMD.EXE        194     BEEPS\admin
pulist.exe     88      BEEPS\admin
```

Then if we need to kill the X Windows Server process (pid 190), we could use the kill command, also include with the Windows NT 4.0, Server Resource Kit.

```
$ rsh beeps '\\ivy\bin\ntreskit\kill 190 '
process #190 killed
```

One final example of utilizing command line tools for monitoring purposes is given in **Appendix A**. This batch file is run on a Windows NT client at boot time via the AutoexNT service included in [**8**] . Each client has been configured to save the memory dump in the case of a system crash ("Blue Screen of Death"). The script, among other things, checks for the existence of the %SystemRoot%\memory.dmp file. If the file is located, an appropriate message is emailed to the network administrators indicating that the machine crashed. If the file does not exist, then it must have been power cycled.

Remote Updates of Systems, Applications and Associated Settings

Without the existence of a true Windows NT PDC, certain tools provided for remote updates of clients, such as Microsoft's System Management Server (SMS), become unavailable. The reason is that such tools often only run under Windows NT and must be installed on an NT PDC or BDC. Alternative solutions for automatically updating remote systems had to be developed. The solution was to utilize a method that had previously been developed for updating UNIX machines. The mechanism is fully explained in [**9**]. Since the publishing of the original Patch32 system, the method has been expanded to deploy applications such as anti-virus software, Netscape Communicator 4.08, and various network connectivity applications.

In addition to deploying updates and applications, the main Perl script was modified to email the logfile of any changes that were applied. In this way, system changes can be logged and monitored (see **Listing 3**).

Listing 3: E-mail message that acts as a log for monitor remote system updates.

```
Subject  : Machine Patched [WinNT] : mepc47
   Date  : Thu, 27 May 1999 17:07:06 –0500
   From  : admin@eng.auburn.edu
     To  : pcpatch@eng.auburn.edu

Installing hostex4...
 \\ivy\patch32\winnt\1381\hostex4\install.inf

Installing the_net...
 \\ivy\patch32\winnt\1381\the_net\install.inf
```

The details of deploying applications via the Patch32 script are beyond the scope of this paper. However, the point of interest are identical to deploying operating system updates and can be gleaned from the previously mentioned paper.

What Have we Learned?

The success with which we have been able to implement a Windows NT domain without a true Windows NT Server acts as proof of concept. It is possible to provide a Windows NT environment for users without great sacrifices to the services that a true Windows NT domain would offer. Of course there are differences, but from the user's point of view these are minimal. After all, it is the user we are attempting to best serve, is it not?

These results are obviously due primarily to the development of Samba and its associated functionality. There are, however, other issues that are not solved simply by the ability to support a domain logon. Other issues such as user management, remote administration and remote updates must be addressed as well. Just as all of these topics can not always be addressed solely by the Windows NT operating system, neither can Samba be the end all be all of an NT domain. Other tools do exist or can be developed with some effort to supply the necessary functionality to completely manage a Windows NT domain. Whether this is the best solution for your network, the reader is advised to consider all the costs and benefits.

Of the five topics covered, only two are tied to Samba and its implementation as a Primary Domain Controller. File and Printer service, remote administration, and remote updates of systems could all be implemented in a pure Windows NT environment using the techniques presented in here.

References and Resources

[**1**] Leighton, Luke Kenneth Casson, "NT 3.5/4.0 Domains for UNIX", Conference proceedings from the First Annual Large Installation System Administration of Windows NT, 1998.

[**2**] "Samba FAQ for NT Domain PDC Support", Available at all mirrors of the Samba web site (http://samba.org)

[**3**] Carter, G., "Linux Rules the Domain", Linuxworld, http://www.linuxworld.com/linuxworld/lw-1999-05/lw-05-thereandback.html, May 1999.

[**4**] Carter, Sharpe, Sams Teach Yourself Samba in 24 Hours, Sams Publishing, 1999.

[**5**] Carter, G., "How to cross the sometimes tenuous bridge between Linux and NT", LinuxWorld, http://www.linuxworld.com/linuxworld/lw-1998-10/lw-10-thereandback.html, October 1998.

[**6**] Visser, J., "On NT Password Security", Open Solution Providers, http://www.osp.nl/infobase/ntpass.html

[**7**] Carter, G. "Command-line NT: It does exist!", LinuxWorld, http://www.linuxworld.com/linuxworld/lw-1999-04/lw-04-thereandback.html, April 1999.

[**8**] Microsoft Windows NT Server Resource Kit, Microsoft Press, 1996.

[**9**] Carter, G., "Patch32 : An System for Automated Client OS Updates", Conference proceedings from the First Annual Large Installation System Administration of Windows NT, 1998.

[**10**] Web page for this paper, http://www.eng.aubun.edu/users/cartegw/non-NT_PDC

Appendix

%SystemRoot%\Systemr32\Autoexnt.bat script

```
rem ***********************************************************
rem ** Environment variables                                 **
set LOGFILE=%SYSTEMROOT%\local\log\autoexnt.log
set LOCAL=%SYSTEMROOT%\local
set ADMINMAIL=cartegw@eng.auburn.edu
set SUBJECT=host %COMPUTERNAME% rebooted

rem ** Timestamp the log file **
echo :::::::::::::::::: > %LOCAL%\temp.log
date /t >> %LOCAL%\temp.log
time /t >> %LOCAL%\temp.log
echo :::::::::::::::::: >> %LOCAL%\temp.log

rem ****** Start the Workstation service ******
%SYSTEMROOT%\system32\net start LanManWorkstation >> %LOGFILE%

rem ****** Update the system clock ******
%SYSTEMROOT%\system32\net time \\ivy /set /yes >> %LOGFILE%

rem ****** Apply any necessary patches ******
if not exist %SYSTEMROOT%\memory.dmp goto patch_os

   if exist %SYSTEMROOT%\local\memory.dmp del %SYSTEMROOT%\local\memory.dmp
   move %SYSTEMROOT%\memory.dmp %SYSTEMROOT%\local\etc\memory.dmp
   echo [%COMPUTERNAME%] : Blue screen! >> %LOCAL%\temp.log
   set SUBJECT=[%COMPUTERNAME%] : Blue screen!
   echo. >> %LOCAL%\temp.log

:patch_os
   \\ivy\perl5\bin\perl \\ivy\patch32\patch32.pl >> %LOCAL%\temp.log
   type %LOCAL%\temp.log >> %LOGFILE%
   echo. >> %LOCAL%\temp.log

type %LOCAL%\temp.log | \\ivy\bin\blat\blat - -t %ADMINMAIL% -f
     roundup@eng.auburn.edu -s "%SUBJECT%" -server mailhost.eng.auburn.edu
del %LOCAL%\temp.log

rem ****** set the chkdsk flag to check the hard disk on reboot
echo y| chkdsk c: /f
```