## CAP SANS GIAC

### Track 5: Securing Windows NT
### Practical Assignment

# Moving secured files to a new server

By Todd Pukanecz

## Introduction

This paper will discuss methods of migrating Windows NT files secured with NTFS permissions, and the shared directories with their permissions, when a sever must move to a new domain or to new hardware. I will give an overview of security identifiers, their relationship to file permissions, and tools for manipulating them. This is presented in the context of a project to move the data from an existing file server to server with improved hardware.

I have used some or all of the techniques described in this paper in actual projects of renaming a Windows NT domain, downgrading a Windows NT domain controller to a Windows 2000 member server in an NT domain, as well as the project of migrating a server to new hardware that is the subject of this paper. These techniques can also be used by administrators who need to collapse a complex Windows NT domain structure prior, or during a transition, to Windows 2000 and Active Directory. While Microsoft has excellent guidelines for collapsing a complex domain structure, their instructions require running Windows 2000 in native mode, which eliminates the ability to keep Windows NT domain controllers. Administrators that may need to run Windows 2000 in mixed mode or otherwise wish to collapse a Windows NT domain structure should find these techniques useful.

## Overview of Project (moving HAYSTACK)

The project involves moving a production file server from a low-end workstation to a rack-mounted server. The server is used exclusively as a file server for a medium security database system. The system is in MS Access, and users run MS Access on Windows workstations that are authenticated through our Windows NT domain. Permissions to the database files are secured through Windows share and NTFS permissions. In addition to hosting the files, the database system requires Oracle client software and ODBC drivers to download data from another server.

The required result of this project is to exactly preserve the files, folders, and shares with their associated permissions. It is desirable to complete the project without purchasing new software. It was also desirable to complete the project as quickly as possible, and with as little administrator intervention as possible.

Since the hardware is entirely different it is not possible to simply take a recent tape backup and restore it to the new hardware. Since there isn't much software required, we decided to reinstall it by hand on the new hardware once the operating system was installed and patched. The crucial issue was the migration of the existing files, folders, and shares with their permissions.

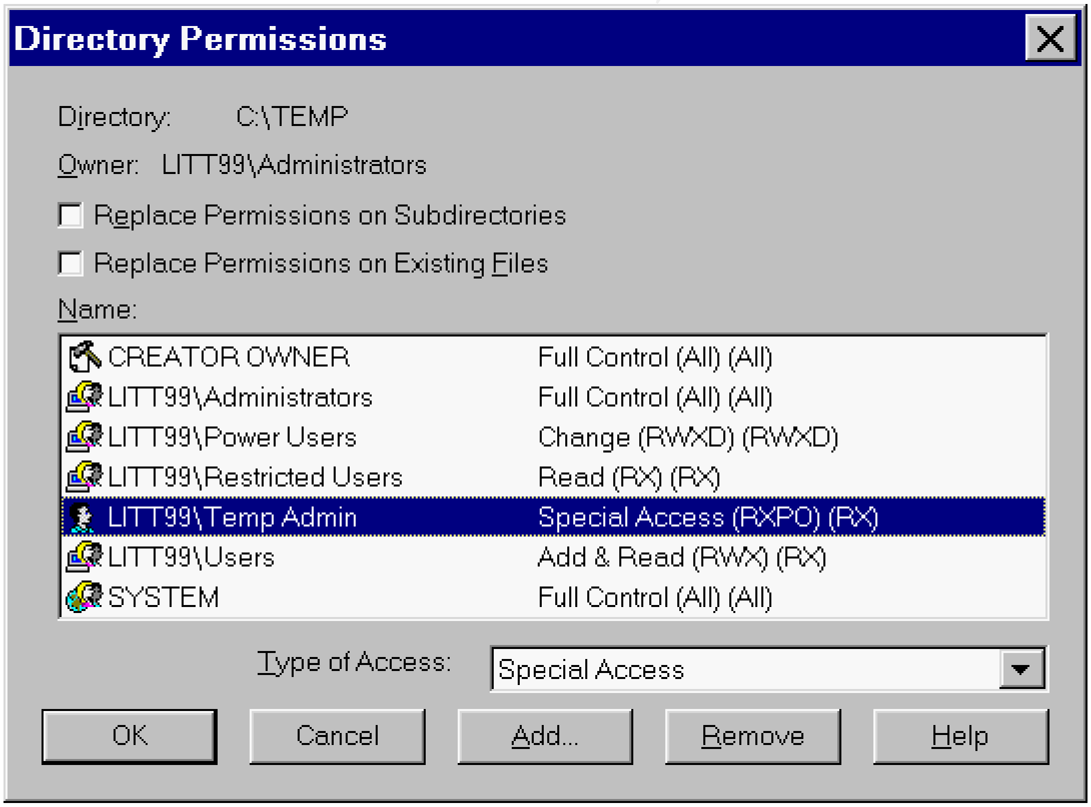
## *Windows NT object access security*

### Overview of Windows NT security model

The Windows NT security model is designed to regulate access to objects on the system. The objects may be files, printers, services, and other shareable or securable devices. The security model identifies each user, group, and accessible object. It regulates access by users to an object, even when the object is accessed indirectly by a program running on the user's behalf. Windows NT can also track successful and failed attempts to access objects.

An administrator or owner assigns permissions to users and groups that grant or deny specific access to an object. The type of access depends on the object. For a file or folder the administrator may grant or deny permission read, write, delete, execute, or modify file permissions. For a printer the administrator may grant or deny permissions to print, manage documents, etc. The ability of the owner or administrator of an object to assign permissions to only specific users is called *discretionary access control*.

Figure 1: Example directory permission dialog box



Directory Permissions

Directory: C:\TEMP

Owner: LITT99\Administrators

□ Replace Permissions on Subdirectories

□ Replace Permissions on Existing Files

Name:

| | |
|---|---|
| CREATOR OWNER | Full Control (All) (All) |
| LITT99\Administrators | Full Control (All) (All) |
| LITT99\Power Users | Change (RWXD) (RWXD) |
| LITT99\Restricted Users | Read (RX) (RX) |
| LITT99\Temp Admin | Special Access (RXPO) (RX) |
| LITT99\Users | Add & Read (RWX) (RX) |
| SYSTEM | Full Control (All) (All) |

Type of Access: Special Access

The above example shows an example of discretionary access control in Windows NT. This is the security dialog for the folder C:\Temp which is displayed by right clicking on the folder, choosing Properties -> Security -> Permissions. In this example, Restricted Users is a new local group and Temp Admin is a new user, both created by an administrator. Administrators is a built-in group that has full control by default. Power Users is a built-in group for users who need more permissions than regular users, but who shouldn't have as much permission as Administrators. Users is another built-in group for ordinary users. SYSTEM is an internal Windows NT account used to authenticate object access for the operating system. This is also the default account used by services such as the task scheduler or the UPS monitor. CREATOR OWNER is another internal Windows NT account that designates the owner of an object.

In this example, an ordinary User (who is not a member of either the Administrators or Power Users group) has Add & Read permission to C:\Temp. If Tommy is such a user he is allowed to create a new file in C:\Temp. When that file is created, Tommy is designated as the owner of the file, and will receive the permissions assigned to the CREATOR OWNER group. Even though Tommy only has Add & Read permission to the folder, he has full control over files he creates. This gives an administrator the flexibility to allow users to create and manipulate new files in a folder without allowing them the ability to alter files created by other users.

## Overview of SIDs

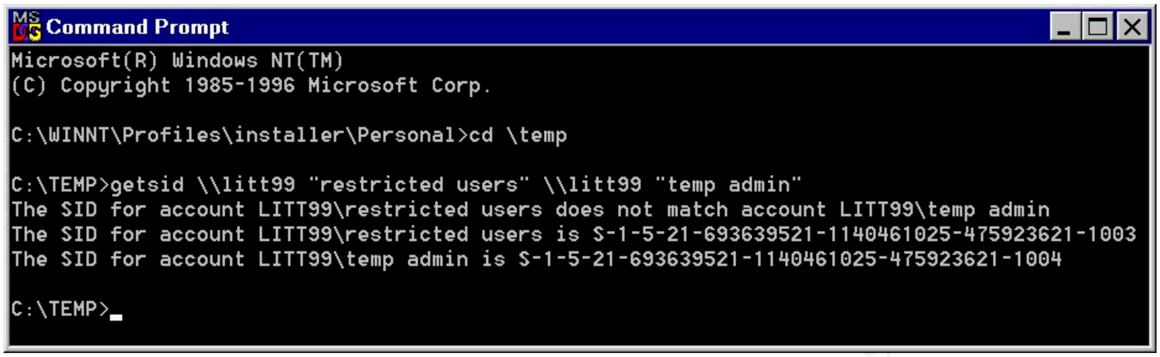
The tools used for manipulating Windows NT permissions present the administrator with a verbal description of the objects they manipulate. Internally Windows uses a *security identifier* to recognize users and groups. The security identifier (SID) is a value that uniquely identifies a user or group. It is generated by the system when the account or group is created.

Each computer in the enterprise is assigned a machine SID. For a Windows NT Workstation, member server, or Primary Domain Controller (PDC) the machine SID is a statistically unique 96-bit number that is computed when Windows NT is installed. For a Windows NT Backup Domain Controller (BDC), the machine SID is the same as for the PDC since all domain controllers share a common account database. The machine SID is prefixed with some identifying information to create the primary SID for that computer or domain. This primary SID is used as a prefix for all accounts and groups created on that computer, concatenated with a *relative ID* (RID) of the user or group to create the account's uniquely identifying SID for the user our group account. All SIDs created on a computer will share the same primary SID and have a RID that will only be assigned once by that computer.

Thus, each SID is unique within the enterprise. A SID for a user or group on a local workstation or member server will not match any SID for any domain users or groups. And a SID in one domain will not match a SID in any trusted domains in the enterprise. The exception to this rule is the built-in users and groups that Windows NT creates on workstations and servers, which are identical on all NT systems.

A SID is unique forever. The Security authorities never issue the same SID or RID twice, and a SID or RID is never reused if an account is deleted. If a user account is deleted and then recreated with the same attributes, it looks identical to the human administrator but has a new RID so it is different to Windows security. Any Windows NT permissions set for the deleted account will not be available to the new account because the SID is never reused.

Figure 2: GetSid output



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\WINNT\Profiles\installer\Personal>cd \temp

C:\TEMP>getsid \\litt99 "restricted users" \\litt99 "temp admin"
The SID for account LITT99\restricted users does not match account LITT99\temp admin
The SID for account LITT99\restricted users is S-1-5-21-693639521-1140461025-475923621-1003
The SID for account LITT99\temp admin is S-1-5-21-693639521-1140461025-475923621-1004

C:\TEMP>_
```

GetSid.exe is a Windows NT Resource Kit (RK) utility that compares and displays the SIDs of users and groups on a workstation or server. The details of GetSid are discussed later, but the output is presented here to illustrate the format of a SID for a user or group, which is:

S-R-X-D-M1-M2-M3-RID

where:

- The letter S which indicates that the value is a SID
- R is the revision level
- X is the identifier authority value
- D is the domain identifier.
- M1, M2, and M3 are the 96 bit unique machine identifier, in three 32 bit sections
- RID is the RID of the user or group

In the above example, the SID for the local group Restricted Users is S-1-5-21-693639521-1140461025-475923621-1003. This SID has

- Revision level 1
- Identifier authority 5, which is Windows NT
- Domain identifier 21
- The machine SID 693639521-1140461025-475923621 which was generated when the Windows NT was installed on the computer
- The relative identifier 1003, which corresponds to Restricted Users on the local computer.

The primary SID for the computer is S-1-5-21-693639521-1140461025-475923621. The RID for the group is 1003. The combination of the primary SID and the RID gives the SID for the group.

In addition to the accounts created by local authorities, Windows NT creates built-in accounts when the operating system is installed. The SIDs for these accounts are:

### Built-In Users

DOMAINNAME\ADMINISTRATOR
(primary SID)-500 (=0x1F4)
DOMAINNAME\GUEST
(primary SID)-501 (=0x1F5)

### Built-In Global Groups

DOMAINNAME\DOMAIN ADMINS
(primary SID)-512 (=0x200)
DOMAINNAME\DOMAIN USERS
(primary SID)-513 (=0x201)
DOMAINNAME\DOMAIN GUESTS
(primary SID)-514 (=0x202)

### Built-In Local Groups

| | | |
|---|---|---|
| BUILTIN\ADMINISTRATORS | S-1-5-32-544 | (=0x220) |
| BUILTIN\USERS | S-1-5-32-545 | (=0x221) |
| BUILTIN\GUESTS | S-1-5-32-546 | (=0x222) |
| BUILTIN\ACCOUNT OPERATORS | S-1-5-32-548 | (=0x224) |
| BUILTIN\SERVER OPERATORS | S-1-5-32-549 | (=0x225) |
| BUILTIN\PRINT OPERATORS | S-1-5-32-550 | (=0x226) |
| BUILTIN\BACKUP OPERATORS | S-1-5-32-551 | (=0x227) |
| BUILTIN\REPLICATOR | S-1-5-32-552 | (=0x228) |

### Special Groups

| | |
|---|---|
| \CREATOR OWNER | S-1-3-0 |
| \EVERYONE | S-1-1-0 |
| NT AUTHORITY\NETWORK | S-1-5-2 |
| NT AUTHORITY\INTERACTIVE | S-1-5-4 |
| NT AUTHORITY\SYSTEM | S-1-5-18 |
| NT AUTHORITY\authenticated users | S-1-5-11 * |

* For Windows NT 4.0 Service Pack 3 and later only

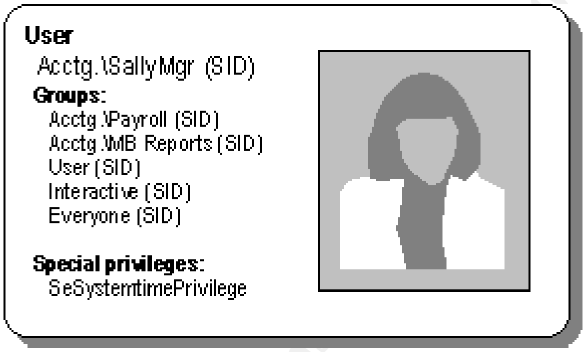
Notice that only the built-in users and built-in global groups include the primary SID of the computer or domain. This is because these are the only objects which are accessible over the network, so are the only objects that can be assigned permissions on other domains. A domain user can access resources on a member server in the domain or in a trusting domain. To avoid confusing the RID of the domain account from the RID of a local account, the appropriate primary SID must be appended to ensure the account is unique in the enterprise.

Built-in local groups and special groups, on the other hand, are never available outside the local computer. I can assign permission or group membership to users and global groups from a trusted domain, but local groups won't cross the domains. Since the built-in local groups and special groups can't be seen over the network they can be the same on all computers. Notice that the built-in local groups use as the domain identification 32 with no machine identifier, which indicates it is a built-in local group. The special groups don't even contain a RID, just the revision, authority, and domain identifier.

## Securing object access

When a user logs on to a Windows NT domain she receives a *security access token*. This token includes the user's SID, the SIDs for groups to which the user belongs, special privileges, as well as other information. Every process that runs on behalf of this user will have a copy of her access token and will use that token to access resources from the application on the user's behalf.

Figure 3: Illustrate the contents of an access token.



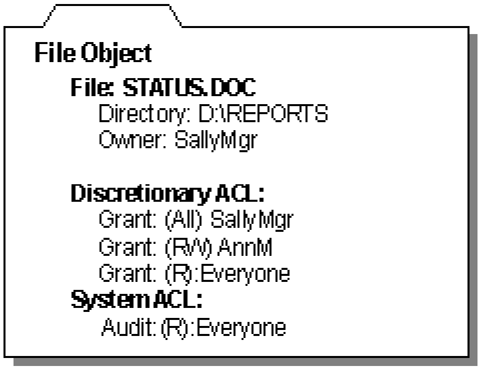
**User**
Acctg.\SallyMgr (SID)
**Groups:**
Acctg.\Payroll (SID)
Acctg.\MB Reports (SID)
User (SID)
Interactive (SID)
Everyone (SID)

**Special privileges:**
SeSystemtimePrivilege

All named objects in Windows NT -- files, folders, printers, etc. -- can be secured. The security attributes for an object are enumerated by a *security descriptor*, which contains the following parts:

- Owner's SID. The owner can change access permissions for an object
- Group SID, serves no purpose outside the legacy POSIX subsystem.
- Discretionary *access control list* (ACL), which associates users and groups with the access permissions granted or denied them. This list is manipulated by the object owner and by administrators.
- A *system ACL*, which controls the generation of auditing messages. The system ACL is controlled by the security administrators.

Figure 4: Example of a security descriptor



**File Object**
**File: STATUS.DOC**
Directory: D:\REPORTS
Owner: SallyMgr

**Discretionary ACL:**
Grant: (All) SallyMgr
Grant: (R/W) AnnM
Grant: (R):Everyone
**System ACL:**
Audit: (R):Everyone

When a user attempts to access an object, Windows NT security authority compares the SIDs in the user's access token with the SIDs on the object's discretionary ACL. The user is granted the permission associated with the SID on the ACL that matches any SID in the user's access token. If there are multiple matches, the user is granted the least restrictive permission. In the illustrated example, Sally Mgr is a member of Everyone, which has only read (R) permission to STATUS.DOC. But because the Sally Mgr account is granted full control (All) over the file, the user of the Sally Mgr account will have unrestricted access to the file. Also note her access to the file will be audited because the Everyone group is audited.

## *Overview of issues for project*

Administrators can control object access and group membership. Administrators can't control SID creation or assignment. We can transfer the existing directory structure complete with access and auditing permissions, and we can recreate the accounts that were associated with those permissions. But because the SIDs for the accounts on the new server will be different, the file permissions will not match the accounts.

This is only an issue for accounts local to the server. Since the new server will be in the same domain, references to domain users and groups will not be a problem. Also references to built-in local groups will not be a problem, since they are the same for all servers. The only problems are for the local accounts, which will have a new primary SID and probably a different RID.

Since we can't control what the SIDs are, we must control which ones are used. What we need is a way to adjust the ACLs on the directory structure to replace the old SIDs with the new SIDs without changing the permission assigned to the SID. Fortunately, the Windows NT Resource Kit has several utilities which will enable the completion of our task.

## *Overview of tools*

These tools are all found in the Windows NT Resource Kit (RK). They can only be used within the context of the licensing agreements associated with the RK. The Resource Kit for any Microsoft product is a treasure trove of utilities, white papers, and other useful resources for an administrator. As a matter of course, we purchase the resource kit whenever we purchase a Microsoft product. Using only the RK tools fulfills the desired result of not having to purchase any more software to complete the project.

### SubInACL.exe

This is the lynchpin of the whole operation. SubInACL will allow us to walk the entire directory tree and replace one SID with another for each file and folder found.

The syntax for SubInACL is:

```
SubInAcl [/view_mode] [/test_mode] [/output=FileName]
             /object_type object_name
             [/action[=parameter] [/action[=parameter]]...

 /view_mode   :
    /noverbose                            /verbose (default=/verbose=2)
    /verbose=1                            /verbose=2
 /test_mode   :
    /notestmode (default=/notestmode)     /testmode
 /object_type :
    /service            /keyreg             /subkeyreg
    /file               /subdirectories     /share
    /clustershare       /kernelobject       /metabase
    /printer            /onlyfile
 /action      :
    /display(default)
    /setowner=owner
    /replace=[DomainName\]OldAccount=[DomainName\]New_Account
    /changedomain=OldDomainName=NewDomainName
    /migratetodomain=SourceDomain=DestDomain
    /findsid=[DomainName\]Account[=stop]
    /suppresssid=[DomainName\]Account
    /confirm
    /ifchangecontinue
    /cleandeletedsidsfrom=DomainName
    /testmode
    /accesscheck=[DomainName\]Username
    /setprimarygroup=[DomainName\]Group
    /grant=[DomainName\]Username[=Access]
    /deny=[DomainName\]Username[=Access]
    /revoke=[DomainName\]Username

Usage  : SubInAcl   [/view_mode] /playfile file_name

Usage  : SubInAcl   /help [keyword]
         SubInacl   /help /full
    keyword can be :
    features  usage syntax sids  view_mode test_mode object_type
    domain_migration substitution_features editing_features
         - or -
    any [/action] [/object_type]
```

As you can see from the object types and actions available, SubInACL is very powerful, allowing a wide range of adjustment to the ACL of files, folders, services, registry keys, shares, and printers. The parameters of interest to us are the object_type /subdirectories, which will walk the entire directory structure making changes, and the action /replace=.

Note that the syntax suggests that the values for the /replace= action should be of the format domain\account. However, since the utility will adjust the SIDs on the ACLs of the files, it must translate the domain\account reference to the appropriate SID. What is not obvious from the syntax description is that the SIDs themselves can be used as parameters of the /replace=. Since permissions are assigned to local groups, and since local groups can't be referenced outside the local security authority, we wouldn't be able to use the syntax /replace=oldserver\account=newserver\account. We would get the error message that oldserver\account couldn't be found.

If we can record the SIDs for the groups on the existing server, recreate the groups on the new server, and record the new SIDs, we can use SubInACL to /replace=oldSID=newSID.

## GetSid.exe

GetSid.exe is an RK utility to display and compare the SIDs for 2 users or group accounts. The account can be local or from the computer's domain or any trusted domain.

The syntax of GetSid is:

```
getsid \\server1 account \\server2 account
```

The output of GetSid is shown in figure 2. We will redirect the output of GetSid to text files to record the SIDs of the existing local groups and users.

## AddUsers.exe

This RK utility is used to enable adding batches of users and groups to a computer. The list of accounts to create is stored in a file, that file is processed by AddUsers, and the accounts are created according to the specifications of the file.

The syntax of AddUsers is:

```
ADDUSERS [/?] [\\computername [[/c | /d | /e] filename]] [/s:?]

  /?    Display this help screen.
  /c    Create accounts specified in the file.
  /d    Write current accounts to the specified file.
  /e    Erase user accounts specified in the file.
  /s:?  Sets the seperator character for the input/output file.
            Replace the ? with the character to be used for seperating
            fields. (eg /s:~)
        Note: The seperator character is a comma ',' by default.
```

Note that AddUsers can be used to not only create accounts from the specifications in a file, it can be used to create a file of account specifications on an existing system. We can use AddUsers to dump the account information on the existing server to a file, then use it to recreate that account information on the new server.

The structure of the file AddUsers requires for creating users and groups is identical to the structure of the file created by AddUsers when it dumps the account information. Since we are using AddUsers at both end of this process, the structure of the file used isn't necessarily important for this project. The file structure is described here for the user's edification:

```
[User]
<UserName>,<FullName>,<Password>,<HomeDrive>,<HomePath>,<Profile>,<Script>
[Global]
<Global Group Name>,<Comment>,<UserName>, ...
[Local]
<Local Group Name>,<Comment>,<UserName>, ...
```

The headings [User], [Global], and [Local] must appear as shown and may have 0 or more entries following them. The [Global] and [Local] groups created may have 0 or more UserNames listed after the Comment, each separated with a comma, and the line must end with a comma or the specified separator character.

Notice that we will be able to transfer most of the attributes of the local user accounts. As a security feature, the password for an account will not be exported by AddUsers. For our project there was only one local user account on the server which is used to run the scheduler service. We determined that it was easier to set a new password for that account rather than try to copy the existing password. For a discussion on dumping and deciphering user passwords see the SANS GIAC NT paper from Monterey 2000 by Brig Otis.

## Scopy.exe

Scopy.exe is a security copying utility. It copies a file, folder, or directory structure along with the existing security descriptor. This allows for copying files, preserving the permissions.

The syntax for Scopy is:

```
SCOPY source destination [/o] [/a] [/s]

source      Specifies files to copy.
destination Where to copy files to.
/o          Copies owner security information.
/a          Copies auditing information.  Requires that you have
            the Manage Auditing User Right on both the source and
            destination computers.
/s          Copies all files in subdirectories.
```

We originally planned to use our tape backup software to restore the files along with their NTFS permissions to the new server. However in testing, we found that using SCOPY from a high-powered workstation ran faster.

## NetDom.exe

NetDom.exe is a command line utility for manipulating domains. It can be used to add a computer to a domain, manage computer accounts for BDCs and member servers, establish trusts with other domains, and manage computer accounts in resource domains.

NetDom has several commands depending on which operation you wish to perform. In our case, I want to add the new member server to the domain after I give it the same name as the old server. The syntax for this command is:

```
NETDOM [/Options] MEMBER [[\\]MemberName] [/Command]

Options are as follows:

   Options                    Description
   ---------------------------------------------------------------------
   /D[OMAIN]:DOMAINNAME       Performs the operation on the primary domain
                              controller of the domain DOMAINNAME.
                              If this option is not used then the domain
                              is the one of which the workstation or the
                              server is a server. If the computer is a
                              domain controller, the operation takes place
                              on the current domain.

   /U[SER]:DOMAIN\USER        User account used to make the connection with
                              the primary domain controller on which the
                              action is to be performed.
                              If this option is not used, the current user
                              account is used.

   /P[ASSWORD]:password       Password of the user account defined along with
                              the option /USER.

   /NOVERBOSE                 Not verbose. Displays only the results of the
                              performed operation.

Commands are as follows:

   Command                    Description
   ---------------------------------------------------------------------
   No command                 Lists the members of the domain. Must be used
                              without any member name.

   /JOINDOMAIN                Joins a domain. Resets the secure channel if the
                              member was already in the domain.

   /ADD                       Adds a computer account for the member.

   /DELETE                    Removes the computer account for the member.

   /QUERY                     Queries domain information of the member.
```

Our strategy is to install the operating system on the new server and give it a temporary name and IP address. Once all the files and permissions have been copied and the local users and groups have been created on the new server, the existing server will be brought down, the new server will be given it's correct IP address and name. NetDom will be used at this point to allow the server to join the domain using the existing computer account.

I originally thought that NetDom would replace the machine SID on the new server with the machine SID from the old server. However, that would mean all the ACLs for all objects on the computer would have to change. The ACL holds the SIDs for the accounts that have access permissions. The SIDs for local accounts includes the machine SID for the server. The local machine SID can't change once it is created.

The NetDom MEMBER /JoinDomain command will adjust the computer account in the domain, replacing the existing machine SID associated with the computer name with the new SID. After this utility is run, the new server will appear to the domain to be the same as the original server, and will function within the domain just as the old server did.

## RmtShare.exe

RmtShare.exe is used to create shares on a remote server. It can also be used to display share information on a server.

The syntax is:

```
RMTSHARE  \\server
          \\server\sharename
          \\server\sharename=drive:path [/USERS:number | /UNLIMITED]
                               [/REMARK:"text"]
                               [/GRANT [user[:perm][ /GRANT user[:perm]]]]
                               [/REMOVE user]
          \\server\sharename=printername /PRINTER [/USERS:number | /UNLIMITED]
                               [/REMARK:"text"]
                               [/GRANT [user[:perm][ /GRANT user[:perm]]]]
                               [/REMOVE user]
          \\server\sharename [/USERS:number | /UNLIMITED]
                               [/REMARK:"text"]
                               [/GRANT [user[:perm][ /GRANT user[:perm]]]]
                               [/REMOVE user]
          \\server\sharename /DELETE

NOTE: if a sharename or path contains spaces, it should be enclosed
          in quotes:
            \\server\"with space"="c:\with space"
```

In this project RmtShare was used only to display and record share information. It could have been used to create the shares on the new server. Used in conjunction with PermCopy.exe, the whole process of recreating the shares can be fairly automated. An outline of using these tools to accomplish share migration is included at the end of the Migrating Shares section of this paper.

## PermCopy.exe

PermCopy.exe will copy permissions from one share to another. The target share may be on the same server or another server.

The syntax is:

```
PERMCOPY \\SourceServer ShareName \\DestinationServer ShareName
```

PermCopy was not used in this project, although it could have been. In the section on migrating shares I give an outline of how the project may have been completed using this utility. For any project more complex than this one, using RmtShare and PermCopy as described in that discussion should be the preferred method. For a sufficiently complex set of share structures it would be worthwhile to create scripts to automate this process further.

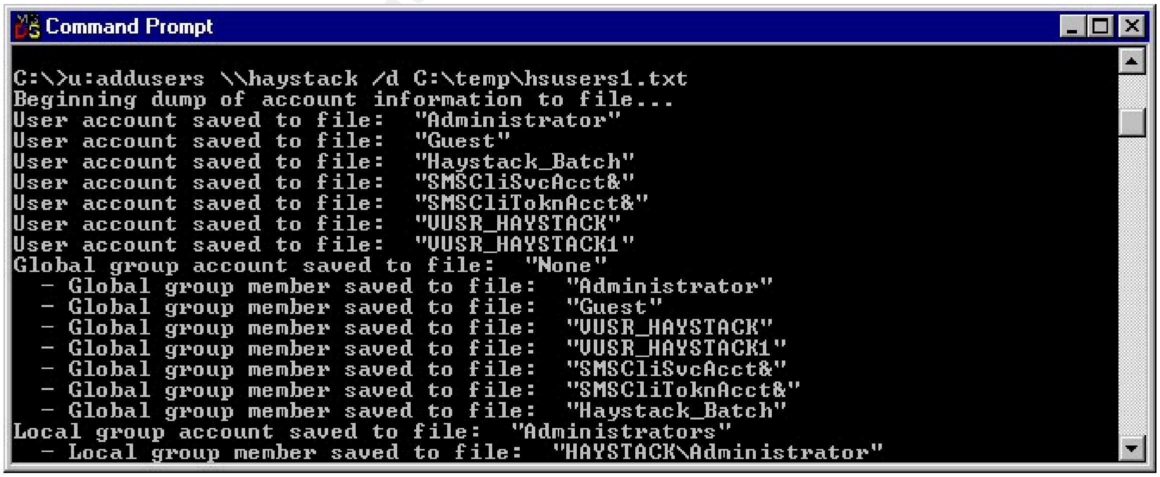
# *Migrating Files*

## Get current information

The first thing to do is save the security information on the existing server. We need to know all the local users and groups, plus their associated SIDs. We also need to save the shares and their associated permissions.

Use AddUsers.exe with the /d parameter to save the information to file, as displayed in figure 5.

Figure 5: Output from AddUsers.exe

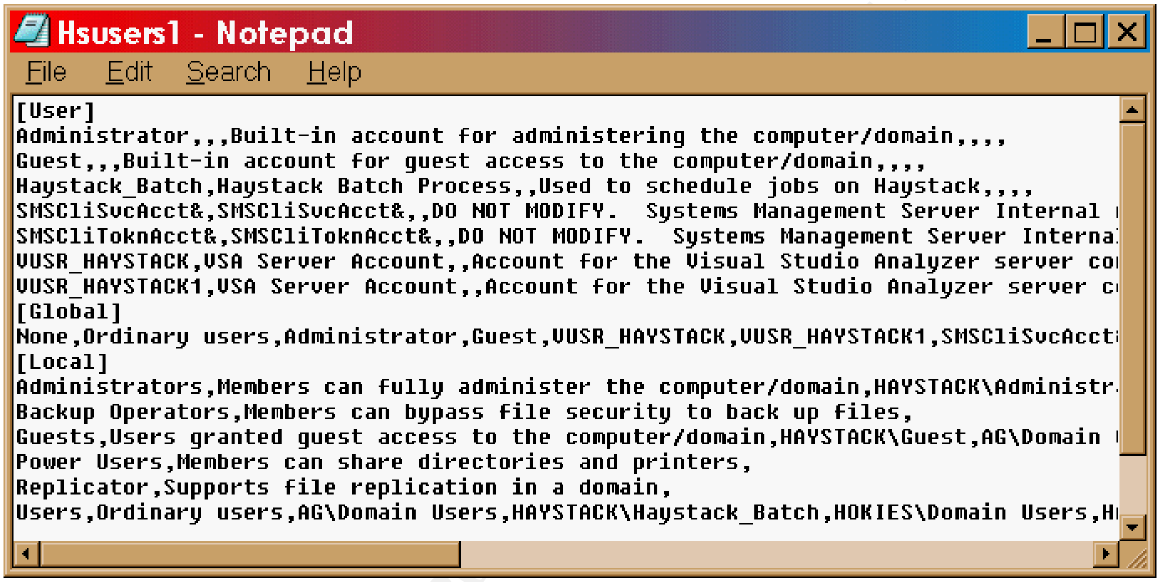


```
C:\>u:addusers \\haystack /d C:\temp\hsusers1.txt
Beginning dump of account information to file...
User account saved to file:  "Administrator"
User account saved to file:  "Guest"
User account saved to file:  "Haystack_Batch"
User account saved to file:  "SMSCliSvcAcct&"
User account saved to file:  "SMSCliToknAcct&"
User account saved to file:  "VUSR_HAYSTACK"
User account saved to file:  "VUSR_HAYSTACK1"
Global group account saved to file:  "None"
  - Global group member saved to file:  "Administrator"
  - Global group member saved to file:  "Guest"
  - Global group member saved to file:  "VUSR_HAYSTACK"
  - Global group member saved to file:  "VUSR_HAYSTACK1"
  - Global group member saved to file:  "SMSCliSvcAcct&"
  - Global group member saved to file:  "SMSCliToknAcct&"
  - Global group member saved to file:  "Haystack_Batch"
Local group account saved to file:  "Administrators"
  - Local group member saved to file:  "HAYSTACK\Administrator"
```

This file will be used on the new server to recreate the users and groups, but it will be edited first. The SMS* users are accounts created automatically by Microsoft Systems Management Server. These will be recreated by SMS on the new server when the SMS client is installed. The VUSR* accounts are created by the database client and again will be recreated when that software is installed on the new server.

The file structure created by AddUsers is fairly straightforward. There are 3 sections, one for users, one for global groups, and one for local groups. In each section the objects are listed, one per line. For users, object attributes like Full Name and Description are listed on the line with parameters separated by commas. For groups, the group description follows the name, followed by the group membership, members separated by commas, as displayed in Figure 6. For this project, editing the file is a simple matter of removing the offending lines.

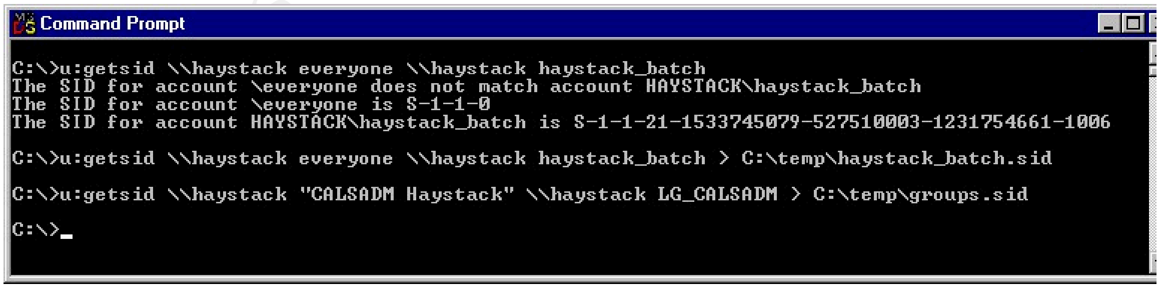
Figure 6: File of accounts created by AddUsers /d



```
[User]
Administrator,,,Built-in account for administering the computer/domain,,,,
Guest,,,Built-in account for guest access to the computer/domain,,,,
Haystack_Batch,Haystack Batch Process,,Used to schedule jobs on Haystack,,,,
SMSCliSvcAcct&,SMSCliSvcAcct&,,DO NOT MODIFY.  Systems Management Server Internal
SMSCliToknAcct&,SMSCliToknAcct&,,DO NOT MODIFY.  Systems Management Server Interna
VUSR_HAYSTACK,VSA Server Account,,Account for the Visual Studio Analyzer server co
VUSR_HAYSTACK1,VSA Server Account,,Account for the Visual Studio Analyzer server c
[Global]
None,Ordinary users,Administrator,Guest,VUSR_HAYSTACK,VUSR_HAYSTACK1,SMSCliSvcAcct
[Local]
Administrators,Members can fully administer the computer/domain,HAYSTACK\Administr
Backup Operators,Members can bypass file security to back up files,
Guests,Users granted guest access to the computer/domain,HAYSTACK\Guest,AG\Domain
Power Users,Members can share directories and printers,
Replicator,Supports file replication in a domain,
Users,Ordinary users,AG\Domain Users,HAYSTACK\Haystack_Batch,HOKIES\Domain Users,H
```

Now save the SIDs with GetSid.exe. Use DOS redirection to pipe the GetSid output to file, as in Figure 7. GetSid requires 2 local objects, and compares the 2 SIDs. Since I had a small local user base, I put 2 SIDs per file. The output of AddUsers from the previous step could potentially be used to automate GetSid for a larger user base.

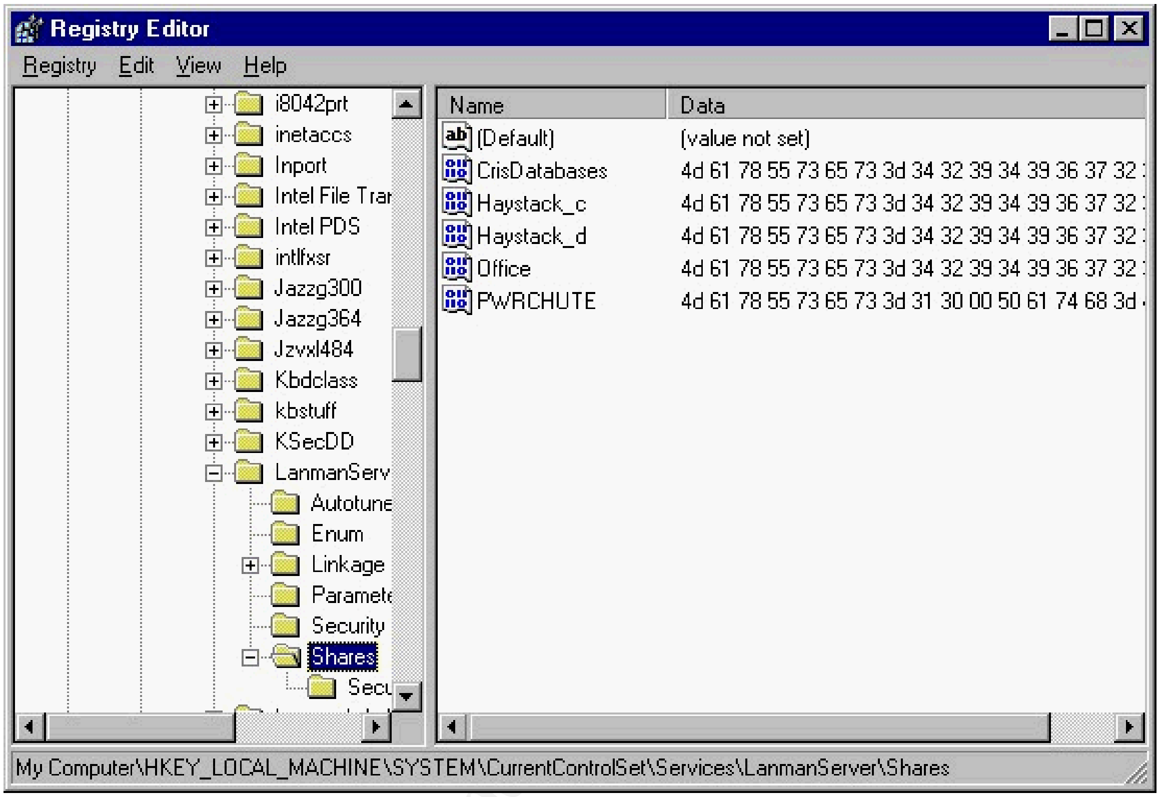
Figure 7: Output of GetSide.exe and using DOS redirection to pipe it to a file



```
C:\>u:getsid \\haystack everyone \\haystack haystack_batch
The SID for account \everyone does not match account HAYSTACK\haystack_batch
The SID for account \everyone is S-1-1-0
The SID for account HAYSTACK\haystack_batch is S-1-1-21-1533745079-527510003-1231754661-1006

C:\>u:getsid \\haystack everyone \\haystack haystack_batch > C:\temp\haystack_batch.sid

C:\>u:getsid \\haystack "CALSADM Haystack" \\haystack LG_CALSADM > C:\temp\groups.sid

C:\>_
```

Share information is stored in the registry at HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Shares, as shown in figure 8. The permissions are stored in the Security key. However, the SIDs on the share ACLs are not stored in a manner which allows manipulation. Since I only have one local group, I can rebuild the permissions by hand. RmtShare.exe can be used to save the share information, as shown in figure 9.

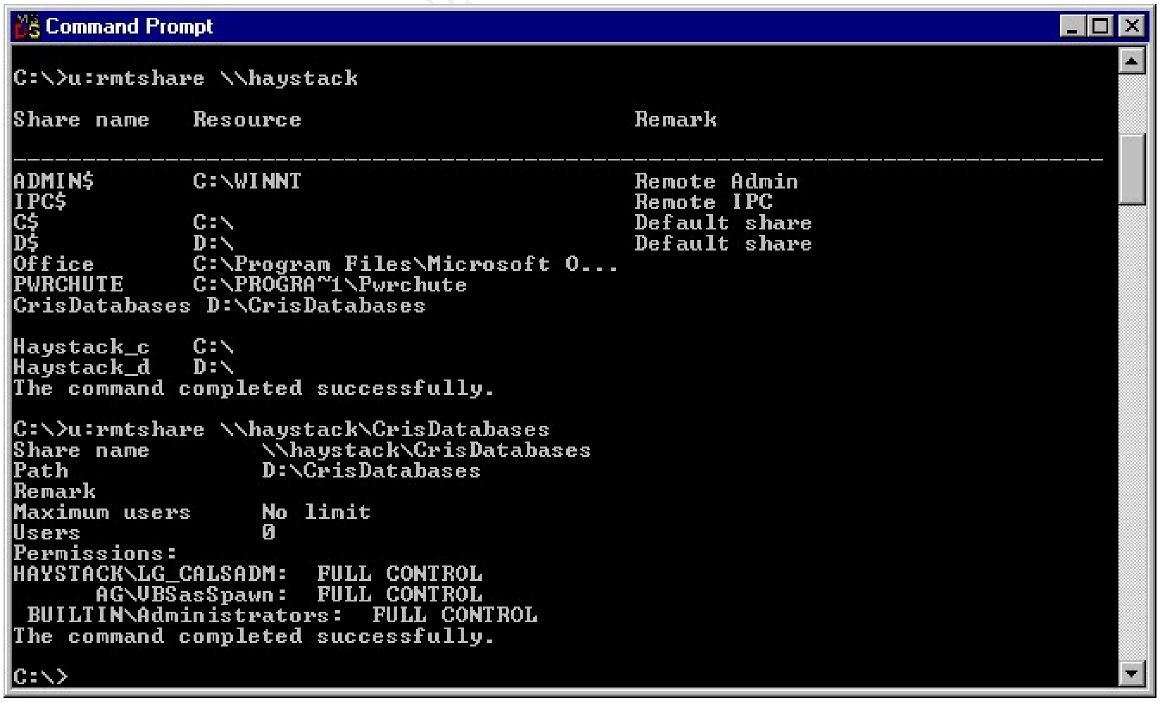
Figure 8: Location in registry of share information



| Name | Data |
|---|---|
| (Default) | (value not set) |
| CrisDatabases | 4d 61 78 55 73 65 73 3d 34 32 39 34 39 36 37 32 |
| Haystack_c | 4d 61 78 55 73 65 73 3d 34 32 39 34 39 36 37 32 |
| Haystack_d | 4d 61 78 55 73 65 73 3d 34 32 39 34 39 36 37 32 |
| Office | 4d 61 78 55 73 65 73 3d 34 32 39 34 39 36 37 32 |
| PWRCHUTE | 4d 61 78 55 73 65 73 3d 31 30 00 50 61 74 68 3d |

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Shares

Figure 9: Using RmtShare.exe to see shares and permissions



```
C:\>u:rmtshare \\haystack

Share name   Resource                        Remark

-------------------------------------------------------------------------------
ADMIN$       C:\WINNT                        Remote Admin
IPC$                                         Remote IPC
C$           C:\                             Default share
D$           D:\                             Default share
Office       C:\Program Files\Microsoft O...
PWRCHUTE     C:\PROGRA~1\Pwrchute
CrisDatabases D:\CrisDatabases

Haystack_c   C:\
Haystack_d   D:\
The command completed successfully.

C:\>u:rmtshare \\haystack\CrisDatabases
Share name         \\haystack\CrisDatabases
Path               D:\CrisDatabases
Remark
Maximum users      No limit
Users              0
Permissions:
HAYSTACK\LG_CALSADM:  FULL CONTROL
      AG\VBSasSpawn:  FULL CONTROL
 BUILTIN\Administrators:  FULL CONTROL
The command completed successfully.

C:\>
```

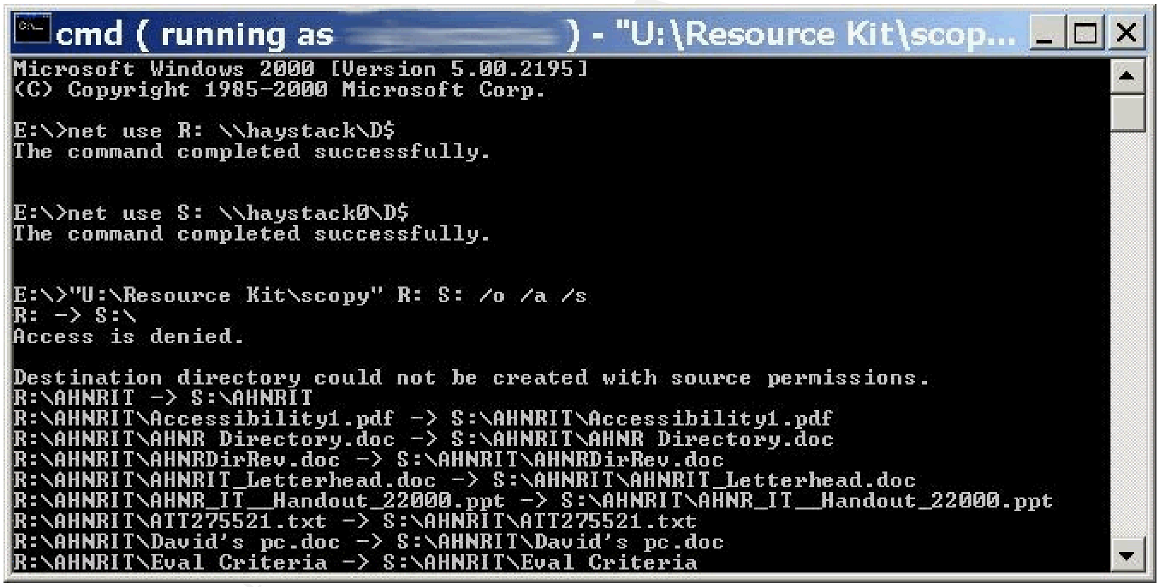
Since working on this project I found the PermCopy.exe utility, which copies a share and the permissions from one server to another. A description of using this utility is included in the discussion on recreating shares.

## The new server

The new server hardware includes a RAID controller for improved reliability. The RAID was partitioned with a system drive and a data drive to match the existing server. Windows NT was installed, a temporary name of HAYSTACK0 was given, the server was made a member server in the domain, and a temporary IP address was applied. Service Pack 6a was applied, the required software was installed, and Service Pack 6a was applied again.

At this point, the new HAYSTACK was an accessible resource in the domain. From an administrator workstation I mapped drives to the two servers and used SCopy.exe to recreate the file and security structure on the data drive, as shown in figure 10.

Figure 10: Using Scopy.exe to copy directory structure with permissions in tact



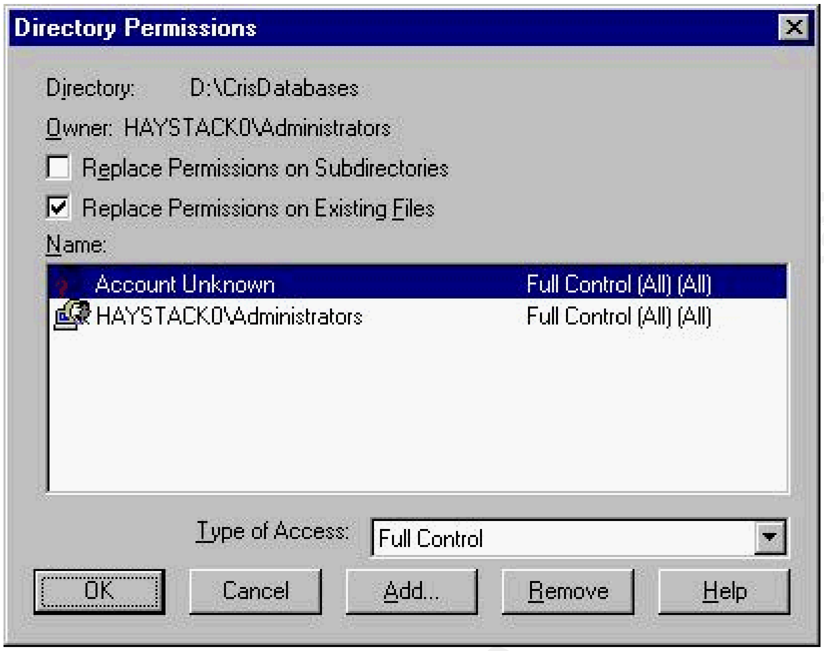
```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

E:\>net use R: \\haystack\D$
The command completed successfully.


E:\>net use S: \\haystack0\D$
The command completed successfully.


E:\>"U:\Resource Kit\scopy" R: S: /o /a /s
R: -> S:\
Access is denied.

Destination directory could not be created with source permissions.
R:\AHNRIT -> S:\AHNRIT
R:\AHNRIT\Accessibility1.pdf -> S:\AHNRIT\Accessibility1.pdf
R:\AHNRIT\AHNR Directory.doc -> S:\AHNRIT\AHNR Directory.doc
R:\AHNRIT\AHNRDirRev.doc -> S:\AHNRIT\AHNRDirRev.doc
R:\AHNRIT\AHNRIT_Letterhead.doc -> S:\AHNRIT\AHNRIT_Letterhead.doc
R:\AHNRIT\AHNR_IT__Handout_22000.ppt -> S:\AHNRIT\AHNR_IT__Handout_22000.ppt
R:\AHNRIT\ATT275521.txt -> S:\AHNRIT\ATT275521.txt
R:\AHNRIT\David's pc.doc -> S:\AHNRIT\David's pc.doc
R:\AHNRIT\Eval Criteria -> S:\AHNRIT\Eval Criteria
```

Notice that the administrator workstation ran Windows 2000 Professional. The RunAs command was used to impersonate the administrator account. For security reasons that account name has been obscured in Figure 10.

Also note that the first message is an Access denied warning and a note that a directory could not be correctly created. This is in reference to the root of the drive. Scopy.exe will try to recreate all files and folders from the starting point listed in the command line parameters. The root always exists on a drive, so we get the message that the destination directory couldn't be created because the root already exists. We get the access denied message because there is no higher level folder in which to create the root.

At this point, the files and permissions have been copied. But, the local users and groups have not been added to the server, so the local accounts are unknown to the server. This can be seen by inspecting the Directory Permissions dialog for the folder CrisDatabases, as shown in figure 11. CrisDatabases gives full control to the local group LG_CalsAdm. However the dialog box shows that the group is unknown to the server.

Figure 11: Directory permissions for D:\CrisDatabases showing "Account Unknown"



Directory Permissions

Directory: D:\CrisDatabases

Owner: HAYSTACK0\Administrators

☐ Replace Permissions on Subdirectories

☑ Replace Permissions on Existing Files

Name:

| | |
|---|---|
| Account Unknown | Full Control (All) (All) |
| HAYSTACK0\Administrators | Full Control (All) (All) |

Type of Access: Full Control

## Rejoin the domain

At this point we have all the information we need from the original server. During a predetermined down period, the original server is taken off-line and the new server is setup to replace it.

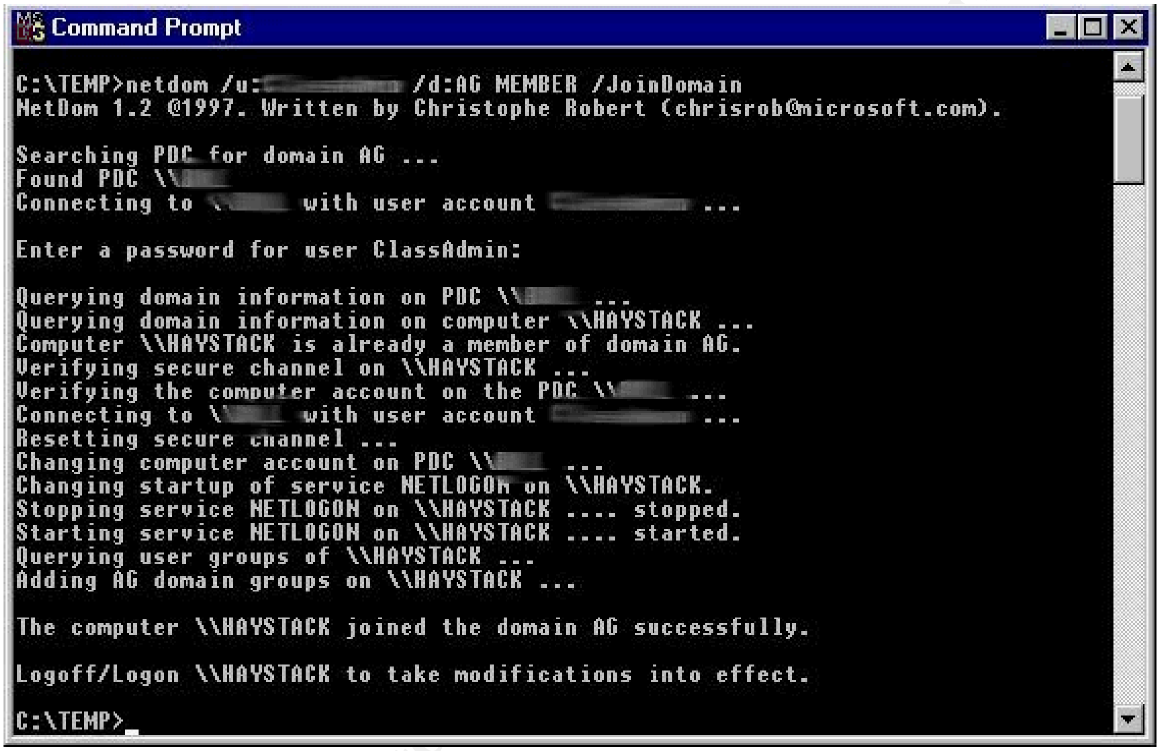
First I replace the temporary IP address on HAYSTACK0 with the IP address of the original server. In Control Panel, use the Network properties. On the Protocols tab select TCP/IP and push the Properties button. Replace the IP address and then push the OK button. Windows NT does not require a reboot after changing IP address.

Before closing the Network properties dialog we can change the NetBIOS name of the server. On the Identity tab, push the Change button. Replace the temporary NetBIOS name HAYSTACK0 with the original name HAYSTACK. Push the OK button, push OK on the warning, push the OK on the Network dialog, and allow the server restart.

There is an existing computer account in the domain for HAYSTACK. The machine SID of the new HAYSTACK does not match the one registered at the PDC. Until this conflict is resolved, network resources will not be available to the server. So, I login to the server using the local administrator account.

Now I can run NetDom.exe to update the machine account at the PDC. I Used the /JoinDomain option of the MEMBER subcommand of NetDom to update this information, as shown in Figure 12. This command is run from the new server which must rejoin the domain. NetDom requires a domain administrator account (or one granted the “Add Server to the Domain” right) to operate. For security reasons this account name has been obscured in Figure 12, as has the name of our PDC.

Figure 12: Using NetDom.exe to rejoin the domain

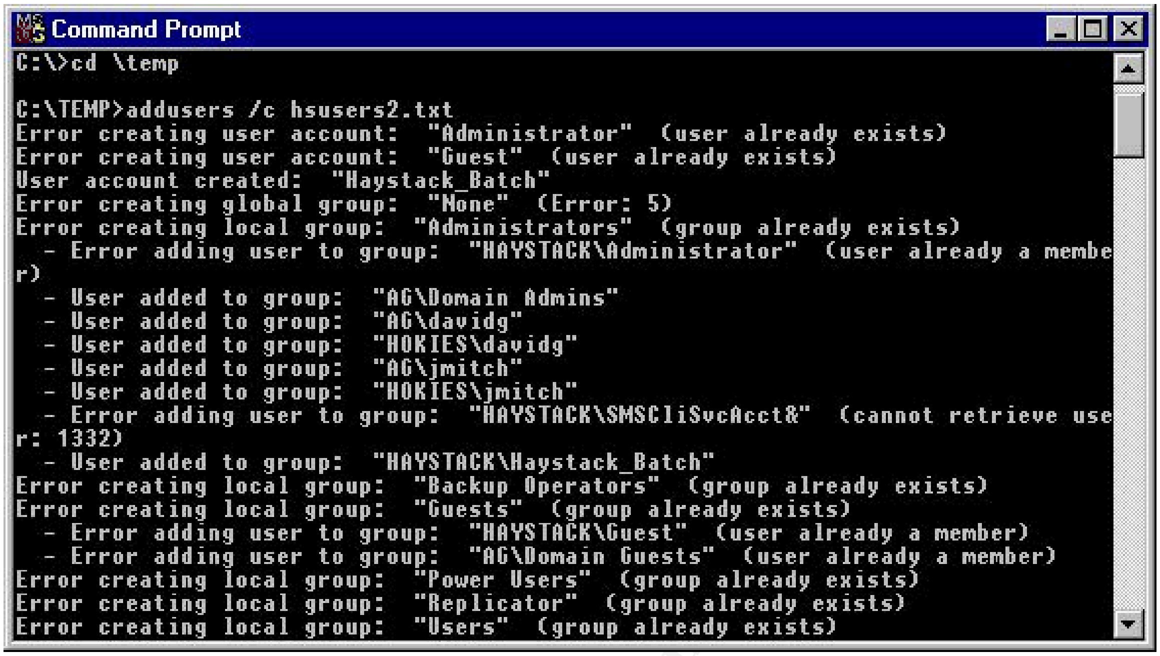


```
C:\TEMP>netdom /u:[redacted] /d:AG MEMBER /JoinDomain
NetDom 1.2 @1997. Written by Christophe Robert (chrisrob@microsoft.com).

Searching PDC for domain AG ...
Found PDC \\[redacted]
Connecting to \\[redacted] with user account [redacted] ...

Enter a password for user ClassAdmin:

Querying domain information on PDC \\[redacted] ...
Querying domain information on computer \\HAYSTACK ...
Computer \\HAYSTACK is already a member of domain AG.
Verifying secure channel on \\HAYSTACK ...
Verifying the computer account on the PDC \\[redacted] ...
Connecting to \\[redacted] with user account [redacted] ...
Resetting secure channel ...
Changing computer account on PDC \\[redacted] ...
Changing startup of service NETLOGON on \\HAYSTACK.
Stopping service NETLOGON on \\HAYSTACK .... stopped.
Starting service NETLOGON on \\HAYSTACK .... started.
Querying user groups of \\HAYSTACK ...
Adding AG domain groups on \\HAYSTACK ...

The computer \\HAYSTACK joined the domain AG successfully.

Logoff/Logon \\HAYSTACK to take modifications into effect.

C:\TEMP>_
```

At the next login, the domain list will be updated and the user may login using accounts in the AG domain or any trusted domain.

At this point I ran AddUsers to import the user base dumped from the original server. This operation is displayed in Figure 13. Note that this step could have been performed at any point between the time we finished installing the operating system on the new server and now.

Figure 13: Using AddUsers.exe to recreate local accounts



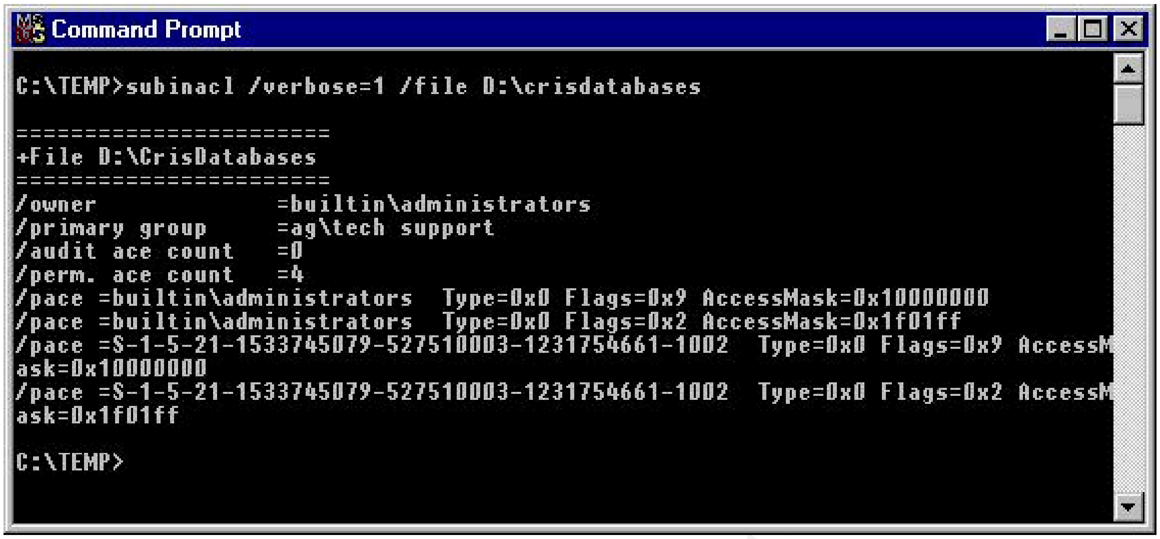
```
C:\>cd \temp

C:\TEMP>addusers /c hsusers2.txt
Error creating user account:  "Administrator"  (user already exists)
Error creating user account:  "Guest"  (user already exists)
User account created:  "Haystack_Batch"
Error creating global group:  "None"  (Error: 5)
Error creating local group:  "Administrators"  (group already exists)
  - Error adding user to group:  "HAYSTACK\Administrator"  (user already a membe
r)
  - User added to group:  "AG\Domain Admins"
  - User added to group:  "AG\davidg"
  - User added to group:  "HOKIES\davidg"
  - User added to group:  "AG\jmitch"
  - User added to group:  "HOKIES\jmitch"
  - Error adding user to group:  "HAYSTACK\SMSCliSvcAcct&"  (cannot retrieve use
r: 1332)
  - User added to group:  "HAYSTACK\Haystack_Batch"
Error creating local group:  "Backup Operators"  (group already exists)
Error creating local group:  "Guests"  (group already exists)
  - Error adding user to group:  "HAYSTACK\Guest"  (user already a member)
  - Error adding user to group:  "AG\Domain Guests"  (user already a member)
Error creating local group:  "Power Users"  (group already exists)
Error creating local group:  "Replicator"  (group already exists)
Error creating local group:  "Users"  (group already exists)
```

Notice the pre-existing groups and users accounts generate an error message. Pre-existing user accounts that already exist in pre-existing groups also generate an error message. AddUsers ignores such errors and continues operating.

## Adjusting Access Control Lists

This is really the heart of the whole operation. At this point we have a valid server in the domain that looks like the original server. The local users and groups have been replicated on the new server. The data files have been copied along with their associated security and auditing information. However, the SIDs of the local users and groups don't match the SIDs of the local users and groups in the ACLs of the data files. This is demonstrated by the output from SubInACL.exe for the folder CrisDatabases as shown in Figure 14. Notice that SubInACL can resolve the SID for the built-in Administrators account, but can't resolve the SID for LG_CalsAdm, so displays the SID instead of the group name.

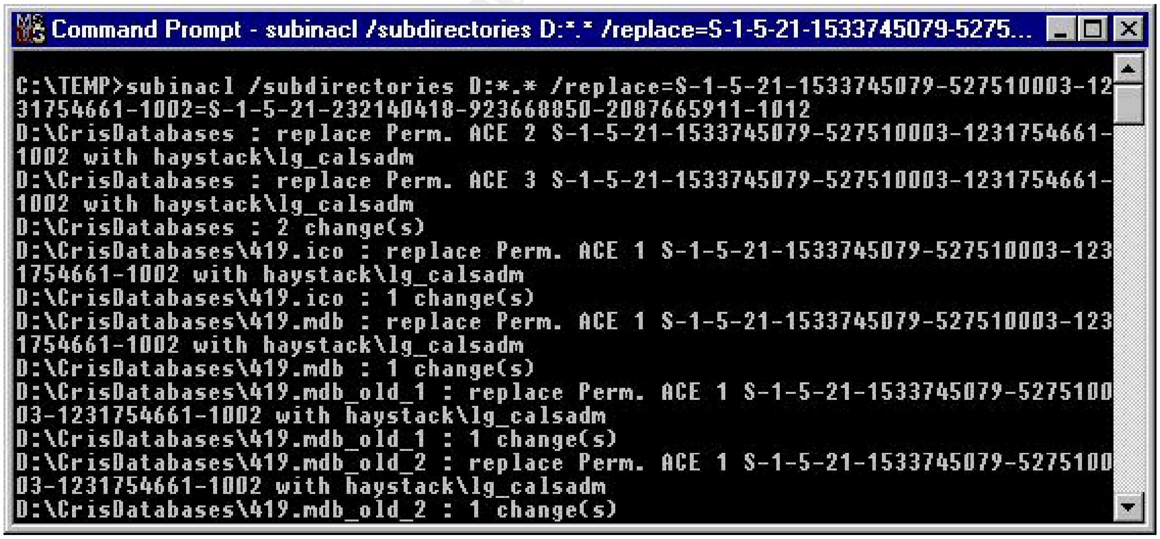
Figure 14: Using SubInACL.exe to show unresolved local group



```
C:\TEMP>subinacl /verbose=1 /file D:\crisdatabases

=======================
+File D:\CrisDatabases
=======================
/owner             =builtin\administrators
/primary group     =ag\tech support
/audit ace count   =0
/perm. ace count   =4
/pace =builtin\administrators  Type=0x0 Flags=0x9 AccessMask=0x10000000
/pace =builtin\administrators  Type=0x0 Flags=0x2 AccessMask=0x1f01ff
/pace =S-1-5-21-1533745079-527510003-1231754661-1002  Type=0x0 Flags=0x9 AccessM
ask=0x10000000
/pace =S-1-5-21-1533745079-527510003-1231754661-1002  Type=0x0 Flags=0x2 AccessM
ask=0x1f01ff

C:\TEMP>
```

I used GetSid.exe to get the SIDs for the recreated local user and group accounts on HAYSTACK. I then used the files created on the original server from that used GetSid to dump the original SIDs for these accounts. I then cut-and-pasted the SIDs into the correct position in the /replace= command, as shown in Figure 15. Use the existing SID first, followed by the recreated SID that must replace it, separating the SIDs with an equal sign. I used the /subdirectories option to have SubInACL work through the entire directory structure, replacing the SIDs as required.

Figure 15: Using SubInACL.exe to replace SID for local group

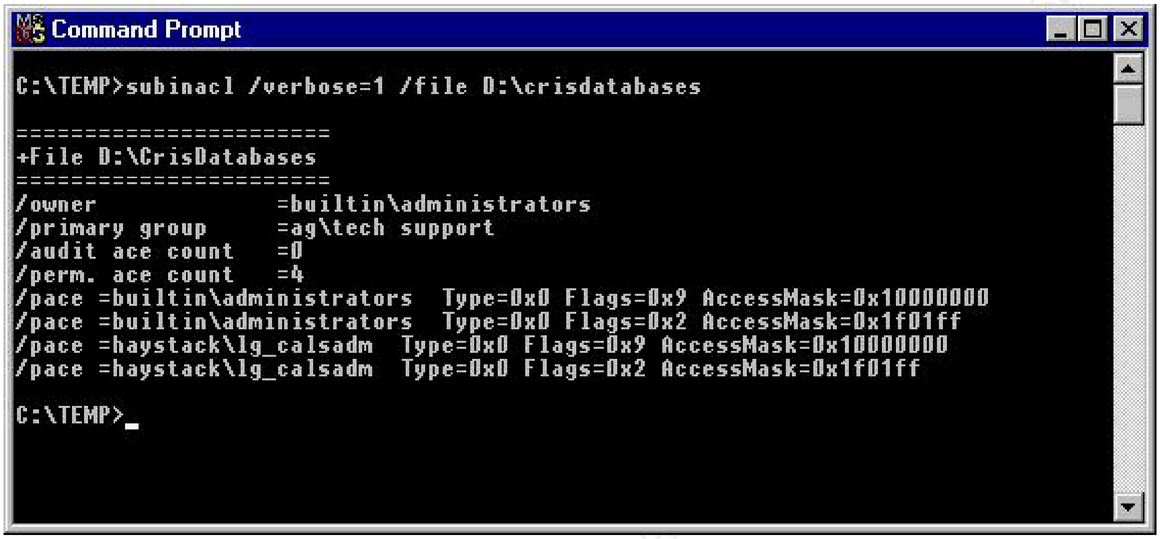

Command Prompt - subinacl /subdirectories D:*.* /replace=S-1-5-21-1533745079-5275...

```
C:\TEMP>subinacl /subdirectories D:*.* /replace=S-1-5-21-1533745079-527510003-12
31754661-1002=S-1-5-21-232140418-923668850-2087665911-1012
D:\CrisDatabases : replace Perm. ACE 2 S-1-5-21-1533745079-527510003-1231754661-
1002 with haystack\lg_calsadm
D:\CrisDatabases : replace Perm. ACE 3 S-1-5-21-1533745079-527510003-1231754661-
1002 with haystack\lg_calsadm
D:\CrisDatabases : 2 change(s)
D:\CrisDatabases\419.ico : replace Perm. ACE 1 S-1-5-21-1533745079-527510003-123
1754661-1002 with haystack\lg_calsadm
D:\CrisDatabases\419.ico : 1 change(s)
D:\CrisDatabases\419.mdb : replace Perm. ACE 1 S-1-5-21-1533745079-527510003-123
1754661-1002 with haystack\lg_calsadm
D:\CrisDatabases\419.mdb : 1 change(s)
D:\CrisDatabases\419.mdb_old_1 : replace Perm. ACE 1 S-1-5-21-1533745079-5275100
03-1231754661-1002 with haystack\lg_calsadm
D:\CrisDatabases\419.mdb_old_1 : 1 change(s)
D:\CrisDatabases\419.mdb_old_2 : replace Perm. ACE 1 S-1-5-21-1533745079-5275100
03-1231754661-1002 with haystack\lg_calsadm
D:\CrisDatabases\419.mdb_old_2 : 1 change(s)
```

Notice that SubInACL can resolve the SID for the accounts on the new server. The output for D:\CrisDatabases shows that the old SID is replaced with the group HAYSTACK\LG_CalsAdm, even though I specified the SID for LG_CalsAdm on the command line. The text name of the group could have been used as a parameter to SubInACL instead of the actual SID.

The output from SubInACL shown in figure 16 is from the same command displayed in Figure 14. Notice that the SID for LG_CalsAdm now resolves correctly and that other data (Flags, AccessMask) did not change.

Figure 16: Using SubInACL.exe to show that local group name correctly resolves.



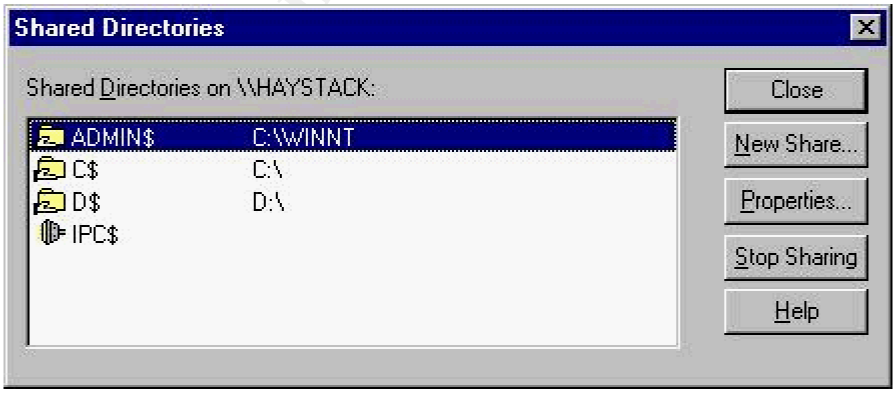
```
C:\TEMP>subinacl /verbose=1 /file D:\crisdatabases

=======================
+File D:\CrisDatabases
=======================
/owner             =builtin\administrators
/primary group     =ag\tech support
/audit ace count   =0
/perm. ace count   =4
/pace =builtin\administrators  Type=0x0 Flags=0x9 AccessMask=0x10000000
/pace =builtin\administrators  Type=0x0 Flags=0x2 AccessMask=0x1f01ff
/pace =haystack\lg_calsadm  Type=0x0 Flags=0x9 AccessMask=0x10000000
/pace =haystack\lg_calsadm  Type=0x0 Flags=0x2 AccessMask=0x1f01ff

C:\TEMP>_
```

## *Migrating Shares*

The share information from the original server was archived at the same time as the NTFS permissions. RegEdit.exe and RmtShare.exe were used, as described on page 15.

Figure 17 shows the Shared Directories dialog available from Server Manager. The only shares available are the default, administrative shares that are created when Windows NT is installed.

Figure 17: Default shares on new server

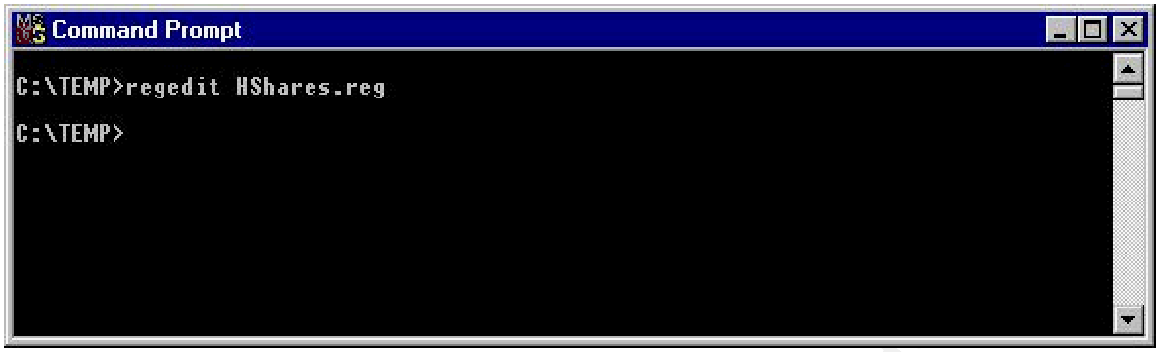


Shared Directories on \\HAYSTACK:

| | |
|---|---|
| ADMIN$ | C:\WINNT |
| C$ | C:\ |
| D$ | D:\ |
| IPC$ | |

Use RegEdit.exe to import the .reg file created on the original server, as demonstrated in Figure 18.

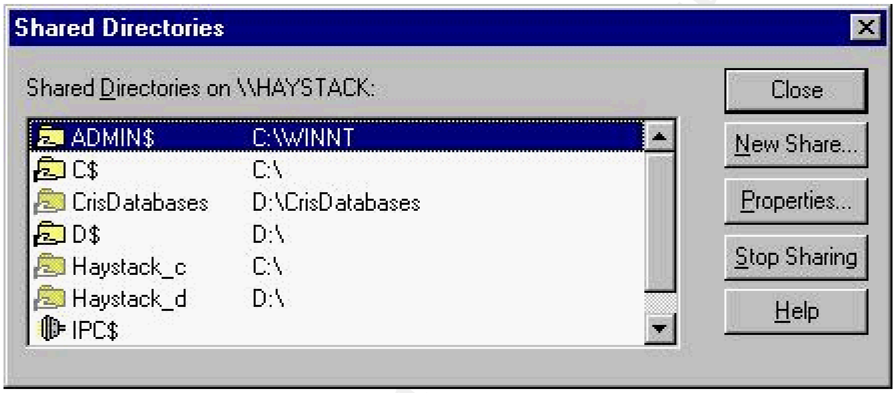
Figure 18: Using RegEdit.exe to import .reg file, which recreate shares



```
C:\TEMP>regedit HShares.reg

C:\TEMP>
```

Figure 19 shows the Shared Directories dialog again, after importing the .reg file with the share information. Notice that all the original shares have been recreated. Also note that the new shares are have a semi-transparent icon. This indicates that the shares have been created but are not yet active. The new shares won't be available until the Server service is restarted, or after the server is rebooted.

Figure 19: Shares created by importing the .reg file

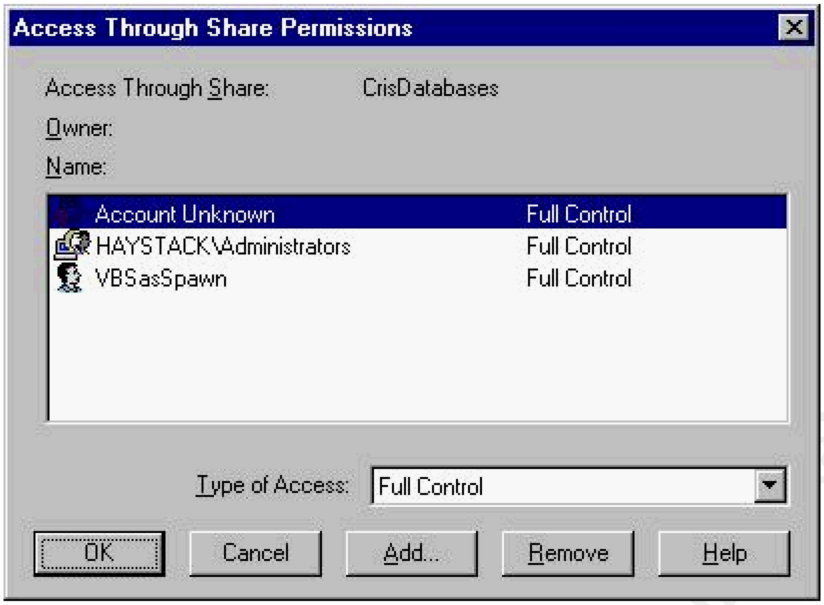


Shared Directories on \\HAYSTACK:

| | |
|---|---|
| ADMIN$ | C:\WINNT |
| C$ | C:\ |
| CrisDatabases | D:\CrisDatabases |
| D$ | D:\ |
| Haystack_c | C:\ |
| Haystack_d | D:\ |
| IPC$ | |

However, we still need to give permissions to the local groups. Figure 20 shows that the local group LG_CALSADM does not yet have permission to the share. Windows can't resolve the SID that has been assigned the permission, and shows "Account Unknown" to indicate this.

Figure 20: Permissions on the CrisData share after importing the .reg file



Access Through Share Permissions

Access Through Share: CrisDatabases

Owner:

Name:

| | |
|---|---|
| Account Unknown | Full Control |
| HAYSTACK\Administrators | Full Control |
| VBSasSpawn | Full Control |

Type of Access: Full Control

Since I only have to give to one local group permission on one of the non-administrative shares, I remove the “Account Uknown” group and add the LG_CALSADM group by hand using the Share Permissions dialog shown in Figure 20.

A more automated approach would have been to use the utility PermCopy.exe to copy the share permissions from the old server to the new server. The rough outline for creating the new server would then be:

1. Install Windows NT, install software, and apply service packs.
2. Run Scopy.exe to copy files along with their ACLs to the new server.
3. run AddUsers.exe to import the local users and groups to the new server
4. Use SubInACL.exe to adjust the ACLs on the files and folders
5. Use RmtShare.exe to display share information on the old server
6. Use RmtShare.exe to create the shares on the new server
7. Use PermCopy.exe to copy the share permissions from the old server to the new.
8. Shut down the old server
9. Adjust the name and IP address of the new server and reboot
10. Login using the local Administrator account and run NetDom.exe to rejoin the domain.

## Conclusions

Microsoft has many tools available for recording and adjusting permissions to Windows objects. To maximize the effectiveness of these tools, it helps to have a good understanding of how objects are represented in the operating system and how they are used to give discretionary access to users. In particular, knowledge of how SIDs are created and how they may be manipulated brought this project to a swift and successful conclusion. Although it would have been rewarding to create scripts to automate more of this process, the time taken to do so would have exceeded the time to perform the operations by hand.

## Bibliography

Fossen, Jason, and Kolde, Jennifer, **Securing Windows NT, Step-by-Step**, The SANS Institute GIAC Training, October 2000.

Western, Ken; et. al., **Windows NT Resource Kit**, Microsoft Press, 1997

Moore, Sonia Marie; et. al., **Windows NT Workstation Resource Guide**, Microsoft Press, 1995, Chapter 6

**Windows 2000 Server Distributed Systems Guide**, Microsoft Press, 1999, Chapter 12

**Building Enterprise Active Directory Services: Notes from the Field**, Microsoft press, 1999

**C2 Administrator's and User's Security Guide**, Microsoft Corporation, 1998,

Otis, Brig, **SANS Practical Track 5: Windows Security, Monterey 2000**