From rpcclient, enumerate user and group settings:

- **`enumdomusers`**: Enumerate users
- **`enumalsgroups domain`**: Enumerate domain groups
- **`enumalsgroups builtin`**: Enumerate local system groups
- **`enumdomains`**: Enumerate domain information
- **`enumprivs`**: Enumerate user system privileges
- **`lookupnames *username*`**: Identify the SID for the username
- **`queryuser *RID#*`**: Identify user information for the given user relative ID number

Create a new user on the remote Windows system using `rpcclient` with the `createdomuser *username*` command.

```
rpcclient $> createdomuser username
rpcclient $> setuserinfo2 username 24
'NewPassword'
```

In this example, the *24* value represents necessary Windows information class constant to set a user password. The value will always be 24 when setting a password.

Create a new share on the remote Windows system using `rpcclient` with the `netshareadd` command.

```
rpcclient $> netshareadd "C:\Windows"
"Windows" 10 "Windows Share"
```

Change a user's password on the Windows system using `rpcclient` with the `chgpasswd3 *username oldpass newpass*` command.

```
rpcclient $> chgpasswd3 josh oldpass
newpass
```

Use `rpcclient` to enumerate Windows password policy information with the `getdompwinfo` and `getusrdompwinfo *RID#*` commands:

```
rpcclient $> getdompwinfo
min_password_length: 5
password_properties: 0x00000000
rpcclient $> getusrdompwinfo 1000
min_password_length: 5
    &info.password_properties:
0xb7d1c734 (3083978548)
   0: DOMAIN_PASSWORD_COMPLEX
   0: DOMAIN_PASSWORD_NO_ANON_CHANGE
   1: DOMAIN_PASSWORD_NO_CLEAR_CHANGE
   0: DOMAIN_PASSWORD_LOCKOUT_ADMINS
   1: DOMAIN_PASSWORD_STORE_CLEARTEXT
   1: DOMAIN_REFUSE_PASSWORD_CHANGE
```

In this output we see that getdompwinfo reveals the minimum password length of 5.
Running getusrdompwinfo followed by a user RID (the first standard user is RID 1000, which can be determined with queryuser) indicates a collection of password settings, including that the server does not enforce a password complexity policy (DOMAIN_PASSWORD_COMPLEX is 0).

SANS INSTITUTE

Accessing Windows systems through the SMB and RPC protocols from Linux systems can be complex. Use this guide as a quick reference to simplify this task.

As an analyst, you may be called upon to interrogate Windows workstation or server systems from a Linux host. This could be following an attack when you need to pivot from Linux to Windows, when you are assessing target systems using Slingshot Linux, or any other situation where you have a Linux terminal and need to access a Windows system.

This cheat sheet covers several tools for collecting Windows system information from a Linux host.

In order to use the `smbclient` and `rpcclient` tools, you will need to authenticate to the Windows target. Specify a username with `-U *username*`. Both tools will prompt for a password. Alternatively, you can specify the `--pw-nt-hash` argument, and supply the NT hash value at the password prompt to conduct a pass-the-hash attack instead.

Use `smbclient` to enumerate a list of file shares:

```
$ smbclient -L ip -U username
```

You will be required to authenticate to the server. Replace *ip* with the IP address or host name of the target system; replace *username* with a valid username (using *domain**username* syntax for a domain user).

By default `smbclient` will connect using SMBv1. If you receive the error message *protocol negotiation failed* when connecting, add the argument -m SMB2 or `-m SMB3` to specify SMBv2 or SMBv3 as the minimum security protocol to use when accessing the server.

```
$ smbclient -L ip -U username -m SMB2
```

You can use this feature to evaluate what the minimum SMB version is for the server:

```
$ smbclient -L ip -U username -m NT1
$ smbclient -L ip -U username -m SMB2
$ smbclient -L ip -U username -m SMB3
```

If all succeed, then the server supports all versions of SMB (including legacy versions, which exposes the server to attack). If one or more fail, the next one that succeeds is the minimum SMB version supported.

Use `smbclient` to retrieve files from a Windows file share, similar to an FTP client:

```
$ smbclient -U username //ip/share
```

After authenticating you will see a `smb :\>` prompt. Use the following commands to navigate and access the file share:

- `ls`: List files and directories
- `cd` *`directory`*: Change to a specified directory
- `get` *`filename`*: Retrieve a file
- `mget file1 file2`: Retrieve multiple files
- `put` *`filename`*: Upload a file
- `mput file1 file2`: Upload multiple files
- `mkdir directory`: Make a directory
- `more filename`: Examine the contents of a text file
- `tar c all.tar`: Retrieve all of the files in the current share directory and subdirectories into a local tar file called `all.tar`
- `exit`: Close the session

All `rpcclient` commands can be run noninteractively with the `-c` argument, allowing you to redirect the output to a file, or pipe to another command-line tool:

```
$ rpcclient -U username ip -c
"enumdomusers" > domusers.txt
```

Connect to a target Windows system through the RPC endpoint to interrogate system information using rpcclient:

```
$ rpcclient -U username ip
```

After authenticating you will see a `rpclient $>` prompt. Here you can issue RPC interrogation commands to retrieve information from the server.

From the `rpcclient` prompt, enumerate Windows system information using the `srvinfo` command:

```
rpcclient $> srvinfo
     10.10.0.1      Wk Sv NT PtB LMB
Sec504Student
     platform_id     :   500
     os version      :   10.0
     server type     :   0x51003
```

From the rpcclient prompt, you can enumerate system information using several of the enum commands. To get a list, type enum followed by the Tab key twice at the rpcclient prompt:

```
rpcclient $> enum<TabTab>
enumalsgroups  enumdomusers
     enummonitors   enumprocs
enumdata  enumdrivers    enumports ...
```