Swinburne University of Technology

Unix for Telecommunications

Doubtfire Submission

Communications - Lab Report 3

Submitted By:
Michael Zdravkovski
101608968
2020/10/17 15:24

Tutor:
Quoc Khanh Le

October 17, 2020

C-LR3: Samba - Lab Report 3

1st Michael Zdravkovski
Bachelor of Computer Science
Swinburne University
Melbourne, Australia
101608968@student.swin.edu.au

I. Aim

This report has been completed to a Credit standard. The Samba labs have the aim of learning the process to deploy a Samba server. This Samba server would then be configured to provide access to Windows hosts through configured shares. These shares will be restricted in access through specified settings. Understanding the service configuration process is crucial to being able to complete this lab.

II. Equipment

For the Samba lab, there was a few key pieces of equipment which were used.

A. RULE Host

The RULE host was used to host the Samba. Samba was provided installed on the host. The RULE host was configured within the Swinburne network, with a host name of rule41.i4t.caia.swin.edu.au, which translates to an IP address of 136.186.230.41. The Swinburne network is able to be reached via a virtual private network client.

B. VPN Connection

A Virtual Private Network connection was established through the Cisco AnyConnect Secure Mobility Client.

C. External Host

The Virtual Private Network connection running the Cisco client was an external host which was outside the Swinburne network. This was able to utilise the Cisco client to authenticate and to the Swinburne network. Using this external host, I was able to connect to RULE host rule41 and access the Samba shares.

III. Method

As per lab handout.

IV. Results

The starting and stopping of the Samba service had to be configured. This is visible below.

```
root@rule41:~ # service samba_server start
Performing sanity check on Samba configuration: OK
Starting nmbd.
Starting smbd.
root@rule41:~ #
```

Fig. 1. The configured Samba service running successfully.

Following this, the configuration of the Samba service itself needed to be defined. The global configuration of the Samba configuration file is below.

```
[global]
   workgroup = MSHOME
   server string = rule41.caia.swin.edu.au
   security = user
   encrypt passwords = yes
   hosts allow = 192.168.2. 127. 136.186.230. 10.1.1. 192.168.216. 136.186.246.
   ntlm auth = ntlmv1-permitted
```

Fig. 2. The global settings in the Samba configuration.

The successfully created accounts within the FreeBSD system are visible below.

```
samba:*:1002:1002:samba:/home/samba:/bin/sh
autocollector:*:1003:1003:autocollector:/home/autocollector:/bin/sh
root@rule41:~ #
```

Fig. 3. The created user accounts samba and autocollector in the output of `/etc/passwd` file

Adding these accounts to the Samba password database was also required, and visible below.

```
root@rule41:~ # pdbedit -L -v
---------------
Unix username:        samba
NT username:
Account Flags:        [U          ]
User SID:             S-1-5-21-3330850158-3463587045-1477329115-1000
Primary Group SID:    S-1-5-21-3330850158-3463587045-1477329115-513
Full Name:            samba
Home Directory:       \\rule41\samba
HomeDir Drive:
Logon Script:
Profile Path:         \\rule41\samba\profile
Domain:               RULE41
Account desc:
Workstations:
Munged dial:
Logon time:           0
Logoff time:          9223372036854775807 seconds since the Epoch
Kickoff time:         9223372036854775807 seconds since the Epoch
Password last set:    Mon, 12 Oct 2020 19:58:02 EST
Password can change:  Mon, 12 Oct 2020 19:58:02 EST
Password must change: never
Last bad password   : 0
Bad password count  : 0
Logon hours         : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Fig. 4. The created samba account in the Samba password database.

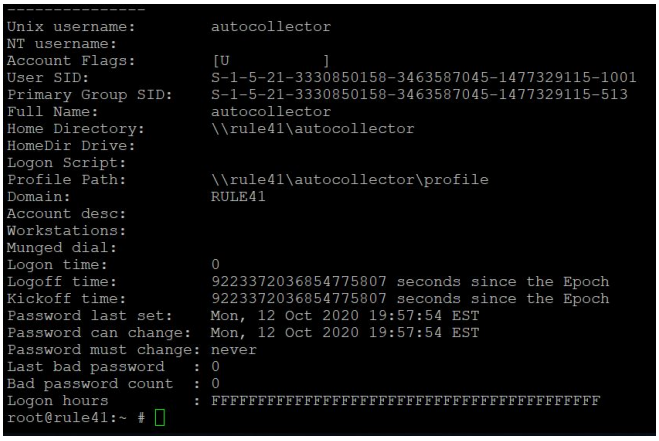

Fig. 5. The created autocollector account in the Samba password database.

Testing the access to the samba share with the samba account was visible below.

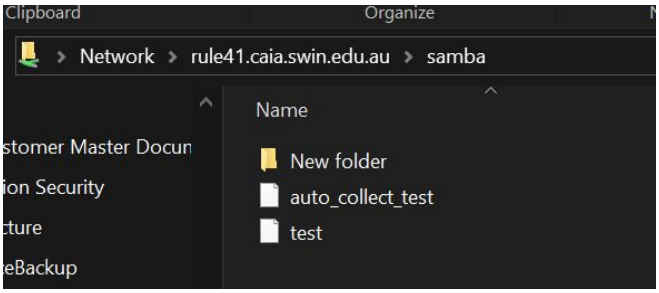

Fig. 6. The samba account accessing the samba share.

Attempting to access the autocollector share with the samba account is visible below.

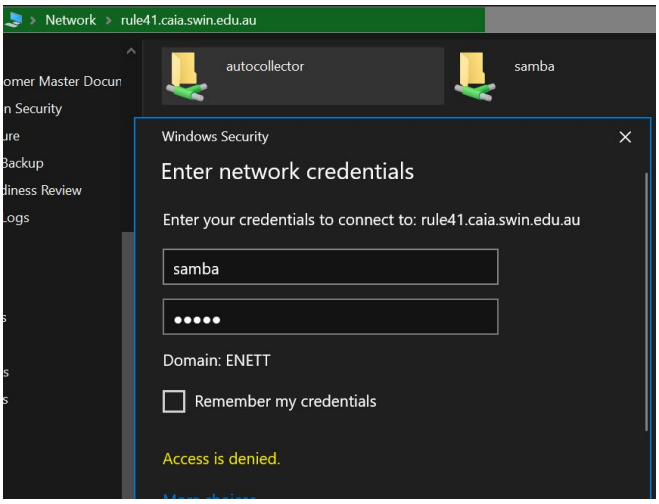

Fig. 7. The samba account failing to authenticate for the autocollector share.

V. Discussion

A. Installing Samba

As Samba was installed locally, there was no need to go through the package installation process. However, this would be a simple process to follow. The typical process would be through the use of the `pkg` command, which allows package names to be specified for installation. The command would be run like so `pkg install packagename`. However, the package would need to be identified. This is also pretty straight forward, as it is a matter of searcing through `pkg search packagename`. Thus, searching for Samba through `pkg search samba` which produces a number of results.

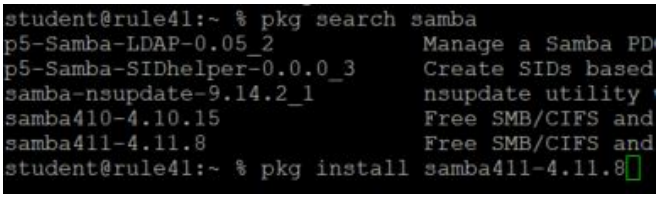

Fig. 8. Searching for and installing the Samba package.

One of the found packages is `samba411-4.11.8`, visible within Figure 8. Thus under FreeBSD it is then a matter of specifying that found package as a parameter into the installation command, `pkg install samba411-4.11.8`.

B. The Samba configuration file

The Samba service will have be configured in order to function and be available to the Windows host which I was using. Understanding this, it was important to be able to identify the location of the Samba configuration file. At first, reading the manual belonging to Samba seemed the logical option. Running the command `man samba` provides the Samba manual. Within the manual, it specified the file being within `/usr/local/samba/lib/smb.conf`, `/usr/samba/lib/smb.conf` or `/etc/samba/smb.conf`.

Being unable to identify the file, I researched the Samba service a bit better. Reading the FreeBSD documentation on Samba, it was a matter of searching for the server configuration regarded. Doing so, it was simple to identify that the server configuration was in `/usr/local/etc/smb4.conf`, as it specifically stated that the ”file must be created before Samba can be used” [1].

C. Starting and Stopping the Samba service

The Samba service is required to be started upon boot of the RULE host. This was pretty easy to find as researching the location of the Samba configuration file also mentioned the line required to be added to `/etc/rc.conf`. It was simply adding `samba_server_enable="YES"`, as seen in Figure 9 below [1].

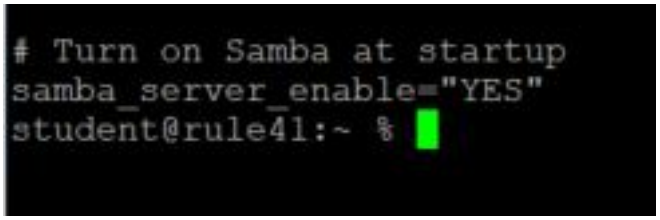

Fig. 9. The rc.conf configuration set to start Samba at startup.

By doing this, it enables the starting and stopping of the Samba service. The results of doing so can be seen in Figure 1. This means not only is the Samba service able to be run via command line, but is also configured to run upon the boot of the RULE host.

D. Configuring the Samba Service

Identifying the location that the Samba configuration file was supposed to be created in earlier allowed for this to take place. The creation of the file `/usr/local/etc/smb4.conf` was of a copy of the provided Samba configuration example as part of the lab. Utilising this, it was a matter of setting the values within the file. The first step was to configure the workgroup to MSHOME, done by adding the line `workgroup = MSHOME`. Then, the configuration required the server string to be changed to the RULE host that was being used, involving the line `server string = rule41.caia.swin.edu.au`. The security mode of Samba was then required, which sets the security level in which Samba operates in. Adding the line `security = user` allowed this to be achieved to a reasonable level. Next, the passwords were required to be encrypted. As this was not provided by the lab instructions, it required reading of the Samba documentation on the Internet.

Upon researching, I found an article stating this was as simple as including the line `encrypt passwords = yes`, as seen within Figure 2 [2]. It specifically highlighted that for this to work, the security mode would need to be in one of "share, user and server", which indicated that this will work for the Samba service as I had configured this to be `user` earlier [2]. The hosts allowed to access the Share shares had to be specified in the configuration. This involved adding the subnet of my IP address belonging to the Windows host from the Cisco VPN adapter, which was `136.186.246.`. This would allow all connections from the VPN subnet to access it, which is important as a VPN connection will never be a static IP from the client. Lastly, NTLMv1 was required to be enabled so that basic authentication could occur with a Windows host. This required research also, and upon reading Samba documentation, it became clear that it needed to be specified. This was done through the line `ntlm auth` being set to `ntlmv1-permitted`, allowing NTLMv1 to be used for "all clients" [3].

E. Setting up Samba Users

Samba requires accounts to be made within the FreeBSD RULE host. This is so that each user can have its home directory shared, and be restricted to only having access to this share. From prior knowledge, it is the `adduser` command which is purposed to create a new user in the FreeBSD system. Running the `adduser` command with no parameters forced the entry of the username, password, home directory, and a number of other parameters which were left as per the defaults. This allowed the samba and autocollector users to be created within the FreeBSD system, with their own distinct home directories of `/home/samba` and `/home/autocollector`. This will later be used to set up the home shares for the two users. This was all visible within Figure 3, where the users were visible within the `/etc/passwd` file, containing information about users on the local Unix system.

Samba also maintains a password database for the users that are required to be accessible via Samba. This was mentioned to be separate to the Unix password database. As mentioned in the lab handout, the command to do so was `smbpasswd`. Reading the manual of this command through `man smbpasswd`, it became clear that running `smbpasswd -a [username]` would allow for an entry within the Samba password database to be created for a specific user. Hence, using the commands `smbpasswd -a samba` and `smbpasswd -a autocollector` prompted for a password to be assigned to the usernames. Usering the password 'samba' for the samba user, and 'autocollector' for the autocollector user, I was able to create the passwords for each user in Samba. This was able to be verified through running the command `pdbedit`. This command is part of the Samba suite, and is used to manage the database of Samba Users, also known as the SAM database. Reading the manual through `man pdbedit`, we are able to see that the flags `-L -v` can be added to list the users with verbosity, displaying their entire entry within the SAM database. Hence, running the command `pdbedit -L -v` provided the output visible within Figure 4 and Figure 5, showing the created users in the Samba password database.

F. Creating the Home Shares

The Samba service also required the configuration of shares, allowing Windows hosts to connect to the path specified and to move files between the two systems. While researching the 'Starting and Stopping the Samba service', the resource found provided a '[src]' example [1]. Using this resource, it became clear how the home shares would have to be configured. First, the share would need to be provided a name, much like the src share had been in the example. The easiest method was to simply name the shares after each user, thus a `[samba]` and `[autocollector]` share was made.

It was clear the each share required the home directory to be accessible, and only the user of that home directory should be able to access it via basic authentication. Identifying this, it was a matter of comparing the example to what our needs were. The user of `path = [home directory]` would be able to specify the home directory of each user. The samba and autocollector users were `/home/samba` and `/home/autocollector`. Then, re-

stricting access to the share was as simple as using `valid users = [username]`. For samba and autocollector, this was `valid users = samba`, and for the autocollector share, `valid users = autocollector`. For each home share, the read/write access must be granted, thus the inclusion of `writeable`, `browsable`, `public` were specified. For both the samba and autocollector shares, this was `public = no`, `writeable = yes`, `browsable = yes`.

```
[samba]
comment = Samba's Home Share
path = /home/samba
public = no
writable = yes
browsable = yes
valid users = samba

[autocollector]
path = /home/autocollector
comment = Auto's Home Share
public = no
writeable = yes
browsable = yes
valid users = autocollector
```

Fig. 10. The smb4.conf file containing the samba shares.

G. Testing Samba

The testing of the Samba service and the permissions established were fairly simple to follow. For both users samba and autocollector, only they must be able to access the share named after their user. They both must have read and write access. Each user cannot access the other's share. As seen within Figure 6, the samba user was able to authenticate to the samba share. The creation of the test file verifies the writing and reading capabilities. The fact that the user had to login, verified the non-public setting of the share. To verify the valid users, it was a matter of samba trying to log into the autocollector share. As visible within Figure 7, it is seen that the samba user is not permitted to authenticate and access to this share. This can be recreated by using a Windows host. Opening the Windows file explorer and entering the name of the rule host in the search bar will enable the traversing to this host. Simply entering `\\rule41.caia.swin.edu.au` should force the user to authenticate as a samba user. This will then display the following:

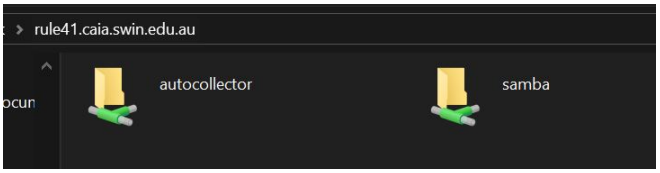

Fig. 11. The Windows file explorer after authenticating as a samba user.

As seen within Figure 11, the two shares should be visible. It is then a matter of attempting to access each. The user which you logged in as should be able to access their home directory, and create/delete files. However, when accessing the other directory, this should not be permitted. A successful authentication to a specific user should display the home directory, as seen within Figure 6. And attempting to login to the other user's share should reject the request, displaying something similar to 7.

Then enter the command `net use \\rule41.caia.swin.edu.au\[username] /delete` to delete the access to the share to be able to authenticate as the other user and test the same.

VI. Conclusion

Utilising Samba has provided access to files stored in Unix to Windows hosts. It allows for a defined access level to be applied, restricting access to only those who should have access to it. As seen throughout the lab, the process is not very in-depth, but requires users to be created in FreeBSD and Samba, and then the shares specified with access for each of those users. This process should be able to be recreated by following the report and will produce a functional Samba service. The testing of the restrictions defined is fairly simple, and should be a matter of authenticating and creating files. Overall, the Samba labs have provided insight into the process of creating a Samba server, and understanding how to properly configure it to not only provide access to shares for specific users, but ensure that it is encrypted and created with a specific security level for users and specified network hosts.

References

[1] "29.10. File and Print Services for Microsoft Windows Clients (Samba) Chapter 29. Network Servers", FreeBSD. Accessed on: October 16, 2020. [Online]. Available: https://www.freebsd.org/doc/en˙US.ISO8859-1/books/handbook/network-samba.html

[2] Blair J, "Samba's Encrypted Password Support", Linux Journal. Accessed on: October 16, 2020. [Online]. Available: https://www.linuxjournal.com/article/2717

[3] "smb.conf", Samba. Accessed on: October 16, 2020. [Online]. Available: https://www.samba.org/samba/docs/current/man-html/smb.conf.5.html