Versuch 5
-
let's dance a **Samba-Fileserver**

Sofern Sie ...

- ✔ zuhause oder in Ihrer WG etc. mit wenig Aufwand einen sicheren, ausbaubaren Fileserver sowohl für Windows- als auch für Linux-Clients betreiben möchten...
- ✔ ohne hohe System- und Energie- und Lizenzkosten….

... dann ist vermutlich ein „*Samba-Server*“ die Lösung!

Die ***Client***-Seite von *„Samba“* kennen Sie ja schon aus dem ersten Versuch – mit dieser kann *unter Linux* auf Windows-Freigaben zugegriffen werden, was bekanntlich wie folgt möglich ist[1]:

- Entweder über den Linux-Dateimanager (z.B. *nautilus*) nach Anzeige der Adresszeile (per Ctrl-L) durch Eingabe in der Form von: *smb://servername/freigabename* *(wonach bei erfolgreicher Verbindung zwecks bequemem Wiederverbinden ein „Lesezeichens“ angelegt werden kann...)*
- Oder per „mounten“ der Freigabe an eine beliebige Stelle in den Verzeichnisbaum (vergleichbar mit dem Virtualbox Shared Folder), entweder manuell (per *'mount -t smbfs ...'* resp. *'mount -t cifs ...'*) oder mittels Eintrag in der Datei */etc/fstab* welche automatisch beim Booten eingelesen wird[2].

In diesem Versuch werden Sie sich der Serverseite von *Samba* zuwenden, welche (da *Samba* freie Open-source ist) kostenlos, zuverlässig und schlank daherkommt und in jeder anständigen Linux-Distribution über den Packetmanager nachinstallierbar ist. Mit etwas Linux-Kenntnisse (abgesehen von ein paar Stolperfallen) ist Samba recht einfach zu konfigurieren. *Samba* (und Linux) wird auch in allen NAS-Servern (Network Accessed Storage) verwendet, wie z.B. den Synology NAS.[3]

Im Folgenden konfigurieren Sie Samba aber selbst von der Pike auf direkt auf dem „Raspi“...

Ein Samba-Server stellt Datei- und auch Druckerfreigabe zur Verfügung. Das unter Windows verwendete Dateifreigabeprotokoll **SMB** (steht für "Server Message Block") wurde in der Version 1.0 ursprünglich von Microsoft im Auftrag von *IBM für dessen **LanManager*** Dateiserver entwickelt. Die Weiterentwicklung (**CIFS, SMB2** und **SMB3**) erfolgte dann durch Microsoft (vgl. https://de.wikipedia.org/wiki/Server_Message_Block).

Samba 3.x konnte bloss die älteren IBM *LanManager* und *Microsoft **Windows NT Server*** emulieren - beide verwenden das Netzwerk-Protokoll "*Netbios"* - wogegen der **Samba 4.x** zudem auch den aktuellen *Microsoft Windows **Active Directory Server*** Mode unterstützt (basierend auf IP Protokolle d.h. UDP und TCP sowie DNS-Integration, LDAP/LDAP-Verzeichnisdienst, Kerberos-Authentifikation und „*Group Policy*“ Funktionalität, …).

Im NAS-Bereich trifft man derzeit beide Samba Releases an, wobei im Home-Bereich meist auf den deutlich komplizierteren *Active Directory Modus* verzichtet wird. Aus dem gleichen Grund verzichten auch wir auf den *Active Directory Server Modus*[4], verwenden aber trotzdem den aktuellen **Samba** Release **4.x**

– Loggen Sie sich entweder per SSH auf Ihre*m Ras*pi ein (unter Linux/OS-X also per '*ssh*' oder aus Windows per '*putty*' . Tipp: vorgängig dem WLAN *mjm987* beitreten – als Loginname haben Sie auf dem Raspi *pi* definiert) … und installieren Sie **auf dem Raspi** *das* Package '***samba***'. *(Die Frage betr. WINS beantworten Sie mit <No>)*
Welcher Samba-Release wurde installiert? (eruierbar per '*samba --version*' oder '*smbd --version*'): .....

Den *Samba*-Serverdienst auf einem Linux-Gastsystem (unter Virtualbox, Parallels etc.) einzurichten wäre zwar prinzipiell auch möglich, jedoch nicht ganz unproblematisch:
Standardmässig richtet die Virtualisierungssoftware nämlich einen ***virtuellen NAT-Router*** auf dem Host-System ein, also **zwischen dem virtuellen Ethernet-Interface des Gast-Systems und dem physikalische Ethernet- oder WLAN-Interface des Host-Systems**. Dieser virtuelle NAT-Router führt *Network-Adress-Translations* durch - genau so wie dies Ihr Router zuhause zwischen dem Heim-LAN und dessen WAN-Interface zum Internet Provider vollzieht.
Damit wäre aber (wie Zuhause) ohne zusätzliche Konfig nur ein Verbindungsaufbau vom Gastsystem *hinaus* in Richtung Internet möglich, jedoch kein Verbindungsaufbau von einen Rechner von ausserhalb über den virtuellen NAT-Router zum Gastsystem, auf welchem der Samba-Serverdienst läuft...[5]

1 s.a. http://wiki.ubuntuusers.de/Samba_Client_cifs
2 Auf Notebooks resp. wenn der Server nicht permanent verfügbar ist, ist diese Variante problematisch, denn das System bleibt beim Bootvorgang hängen, sofern der Dateifreigabe-Server nicht erreichbar ist.
3 Network Accessed Storage: günstige NAS-Server (z.B. von *Synology*) mit intern anschliessbarer SATA-Harddisk(s) oder für einfache Zwecke auch bei manchen NAT-Routern via externe USB-Disk (z.B. die *Fritzbox*).
4 Ein Howto für den *Active Directory* Mode finden Sie hier: https://wiki.samba.org/index.php/Setting_up_Samba_as_an_Active_Directory_Domain_Controller

Im unserem Testszenario verwenden wir deshalb nicht unser Linux-Gastsystem sondern das Raspi als Samba-Server. **Alle nachfolgenden Linux-Commands geben Sie also auf der Raspi-Console über eine ssh-Verbindung** (also via *putty.exe* oder auf OS-X oder Linux per *ssh*) **ein!**
Das Raspi sollten Sie per SSH über dessen Hostname *raspberrypi-XX* (mit XX=Nummer des im Labor verwendeten Raspis) erreichen (oder falls dies nicht klappt mit Anhängen von ".local" also *raspberrypi-XX.local per mDNS)*.
Sofern auch dies nicht klappt, verwenden Sie die IP-Adresse, welche Sie z.B. mittels der Handy App "*Fing*" oder über die asynchron serielle Console eruieren,

– Verifizieren sie auf dem Windows-Client, wie der "**Arbeitsgruppenname"** ihres Windows-Systems lautet (eruierbar auf Windows 10) per: (***Maus-Rechts) Start → System → [Suchen]*** *nach* ***"Arbeitsgruppe"***)

Arbeitsgruppenname: .....

Samba Konfigurationsdatei *smb.conf*

Der Samba-Server wird in einer einzigen Datei konfiguriert: ***/etc/samba/smb.conf***

– Die installierte Original-Datei ist mit sehr viel Kommentar überladen weshalb wir diese wie folgt „retten" und mit einer leeren Konfigurationsdatei beginnen:

```
$ cd /etc/samba
$ sudo mv smb.conf smb.conf.ori
$ sudo nano /etc/samba/smb.conf
```

Wenn das Umbenennen funktioniert hat, öffnet sich eine leere Datei.

Die Grundstruktur von *smb.conf* ist einfach: unter in eckigen Klammern angegebenen **[*Abschnitten*]** werden Einstellungen in der Form ***option = wert*** angegeben.
Detaillierte Erklärungen zu den Optionen finden Sie per:
$ **man smb.conf** oder viele Beispiele in der geretteten Original-Datei *smb.conf.ori*

Unter Abschnitt ***[global]*** werden demzufolge die generellen Server-Einstellungen definiert:

– Erstellen Sie folgenden Abschnitt wobei Sie statt „*Arbeitsgruppenname*" jener Ihres Windows-Clients angeben:

```
[global]
    workgroup = der oben ermittelte Arbeitsgruppenname
    dns proxy = no
```

Der nach aussen sichtbare *Name* des Samba-Servers (im Windows Jargon „*netbios name"*) heisst standardmässig gleich wie der Hostname des Linux-Rechners (vgl. Kommando '***hostname***',) also: .....

– Standardmässig wird als "*Netbios name*" (also als Servername) der Hostname welcher unter */etc/hostname* definiert ist verwendet. Diesen haben Sie ja auf *raspberrypi-XX geändert (*mit XX=Nummer Ihres im Labor verwendeten Raspi).
Dieser "Netbios Name" könnte unter *[global]* mittels einem Eintrag '**netbios name = *was-auch-immer***' umdefiniert werden, was Sie aber besser unterlassen, d.h. falls Innen der Name raspberrypi-XX nicht gefällt, ändern Sie besser den Hostnamen via Ändern von */etc/hostname* und booten das Raspi danach.
Achtung: der *netbios name* und damit auch der Hostname darf max. 14 Zeichen lang sein und keine Sonderzeichen wie '#' enthalten!

– Nach diesem minimalistischen *[global]* Abschnitt können in weiteren Abschnitten noch die gewünschten Netzwerkfreigaben definiert werden: für's erste soll bloss eine Freigabe namens „*temp*" eingerichtet werden, welche das Linux-Verzeichnis ***/tmp*** freigibt:

```
[temp]
    path = /tmp
```

Achtung: Der Freigabename ist im Beispiel "temp", der freigegebene Ordner hingegen bloss der im Root-Verzeichnis bereits bestehende Ordner "tmp" (also ohne 'e' geschrieben!)

5 Realisierbar wäre ein Verbindungsaufbau „von Aussen nach Innen" entweder per „Port-Forwarding" auf dem (virtuellen) NAT-Router, was aber wiederum im Konflikt mit dem Windows Hostsystem stehen würde oder alternativ, indem das virtuelle Ethernet Interface von „NAT" auf „Netzwerkbrücke" (engl. *bridge*) umgestellt wird und somit das Gastsystem die IP-Adresse direkt (per DHCP) vom äusseren Netzwerk (in unserem Fall vom FHNW-Netzwerk) beziehen würde (zusätzlich zu jener des Windows-Hosts). Am FHNW Netzwerk wird dies aber absichtlich verhindert, da nur max. eine IP-Adresse pro WLAN- oder Ethernet-Interface zugelassen wird...

- Nach Speichern der Datei prüfen Sie diese Konfiguration */etc/samba/smb.conf* auf korrekte Syntax per:
  $ **testparm**
  Worauf Sie dann nochmals die *enter*-Taste betätigen, erst dann werden auch die Freigaben angezeigt!
  → **Achten Sie sich auf eventuelle Fehlermeldungen!!!**
- Welche „***Server role***“ wurde dabei gemeldet?.....

Nach **jeder Änderung** von **smb.conf** muss "Samba-Server" **neu gestartet** werden!
Genau genommen besteht der **„Samba-Server“** im gewählten Modus aus zwei Netzwerkdiensten:

nmbd dient (nur) der **Rechner-Namensauflösung** wozu dieser „*Service*“ via **UDP Port 137** und **138** einerseits periodisch den *eigenen so genannten „Netbios-Namen“* auf allen aktiven IP Interfaces „*broadcastet“,* andererseits horcht nmbd auf diesen UDP Ports auch und erfährt dadurch die auf dem lokalen Ethernet-Segment vorhandenen Windows-Rechner (aufgrund deren Broadcasts).

Im (in unserer Konfiguration nicht verwendeten) **Active Directory Modus** erfolgt hingegen die Namensauflösung über einen DNS-Server, welcher hierzu dynamische DNS-Updates zulassen müsste. Da diese auf Heimnetzwerken nicht ganz einfach ist, verzichten wir auf diesen Modus).

Der zweite Dienst ist der ***smbd*** Daemon welcher die eigentliche Dateifreigabe via ***smb***- und ***cifs***-Protokolle zur Verfügung stellt, und zwar einerseits über **TCP Port 445** sowie über **TCP Port 139** („*Netbios over TCP*“, *Netbios* war ein Protokoll aus dem „DOS-Zeitalter“).

- Statt nach jeder Änderung neu zu booten, können die beiden Dienste mittels '*systemctl*' neu starten[6]:
- **sudo systemctl restart smbd** sowie **sudo systemctl restart nmbd**
- Kontrollieren Sie noch per *'ps -ef | grep mbd'* ob die beiden Samba-Prozesse auch wirklich laufen.
- Versuchen Sie nun auf dem Windows-Host auf Ihren Rechner zuzugreifen, per (Maus-Rechts) ***Start→Ausführen: \\servername*temp** (als *servername* den '*hostname*' ihres Raspi angeben!)
  - Erscheint bloss eine Fehlermeldung "***Auf \\servername\temp konnte nicht zugegriffen werden***" (oder ähnlich), versuchen Sie es nochmals mit anhängen von ".local" an den Servernamen also z.B. "*raspberrypi-XX.local*" und falls dies auch nicht klappen sollte, mit der IP-Adresse Ihres Raspi statt mit dessen "*servername" (vermutlich blockt in diesem Fall der Windows-Firewall die Netbios Broadcasts des nmbd Dienstes oder die DNS-Registrierung.)*
  - Nach erneutem Versuch sollte nun zumindest ein Anmelde-Dialog nun erscheinen, **das Anmelden ist aber immer noch nicht möglich…** (s. nächster Abschnitt) [7]

Samba-User(s)

Die Freigabe „*temp“* ist zwar offensichtlich erreichbar, zur Anmeldung fehlt aber noch ein „**Samba-User**“!

Die vorhandenen Linux Passwort-Hashes (in */etc/password* resp. */etc/shadow*) sind nämlich **inkompatibel zum SMB- resp. Windows Freigabe-Protokoll**! Zwar wäre ein anonymer Zugriff auf Freigaben über einen Gast-User per *smb.conf* Option '*guest ok=yes*' möglich, aus Sicherheitsgründen sollten man darauf aber verzichten!

- Erstellen Sie wie folgt ein Samba-User inkl. Passwort, wobei Sie für *username* Ihr Linux-Login angeben:

  $ sudo smbpasswd -a *username*

Achtung (!!): Der angegebene *username* muss immer **sowohl** als Samba-User **wie auch** als Linux-User existieren, denn ***Samba „mapped“* bei Zugriff über die *smb*-Freigabe den vom Client angegebenen Samba-Username auf den gleichnamigen Linux-User**! Verwenden Sie also für *username* **'pi'**

Nun sollte endlich ein Zugriff auf die Freigabe *„temp“* möglich sein, Windows-seitig also z.B. per:

Start > Ausführen: ***\\hostname-ihres-raspi\temp*** oder über dessen IP per: ***\\ip-addresse-ihres-raspi\temp***

6 Der Command *systemctl* ist Teil des Init Systems *systemd*. Zwecks Rückwärtskompatibilität könnte man den smbd und nmbd Dienst auch über die entsprechenden *System-V Init-Scripts* restarten per: '*sudo /etc/init.d/smbd restart*' …

7 Erscheint danach eine Fehlermeldung, dass Sie "*eventuell keine Berechtigung haben, auf diese Netzwerkressource zuzugreifen*", weigert sich Windows die Authentifizierung via das alte *Windows NT/ IBM LanManager* Protokoll durchzuführen. Durch zufügen folgender Gruppenrichtlinie (engl. Group Policy) kann dies behoben werden, per Start des Gruppenrichtlinieneditors per Maus-rechts auf: *Start > Ausführen: gpedit.msc* und darin...
Computerkonfiguration > Administrative Vorlagen > Netzwerk > LanMan-Arbeitsstation → Unsichere Gastanmeldung aktivieren → (x) Akt.

- Überprüfen Sie: standardmässig wird die Freigabe **nur zum Lesen** freigegeben!
- Natürlich kann auch **von der Linux Oberfläche** auf Windows-kompatible Freigaben zugegriffen werden. Versuchen Sie es aus dem Linux Gastsystem aus dem Dateimanager (*nautilus*) wobei Sie zuerst per ***Ctrl-L*** die Adresszeile anzeigen lassen und dann folgendes eingeben:

  smb://*servername*/temp/ oder gleich mit Angabe Ihrer *userid*: smb://*userid@servername*/temp/

  Abhängig von der Art der Namensauflösung (via Netbios Broadcast oder via DNS) funktioniert dies nicht sondern nur via IP-Adresse statt dem Hostname.
  Funktionierte es bei Ihnen über den Hostnamen und wenn ja, welche Art Namensauflösung wurde offensichtlich verwendet?

.....

Defaultmässig arbeitet ein Samba 4.x Server wie bei '*testparm*' angezeigt in der „**server role= *standalone***“ In diesem Mode verwaltet/speichert der Samba Server selbst die samba-Usernamen und samba-Passwörter und benutzt diese auch nur für den Zugriff auf die eigenen Server-Freigaben.
Mit Zuweisung einer der folgenden „**server role**“ könnte Samba 4.x auch:

- Ein „**Member Server**“ sein, in welchem Fall er die Überprüfung der User-Logins einem anderen im Netz existierenden Active Directory Server (Windows oder Samba!) überlässt[8]
- Oder gar ein vollwertiger „**Active Directory Domain Controller**“ sein (für AD Domain-Clients)
- Oder zwecks Kompatibilität zu uralten Systemen auch ein **„NT4 Domain Controller“** emulieren.

Freigeben von Home-Verzeichnissen

Um auf einen Schlag alle existierenden Linux-Homeverzeichnisse freizugeben, ist nur folgender Abschnitt nötig:

```
[homes]
        valid users = %S
        read only = no
        browseable  =  no
```

Damit wird also automatisch das existierende Homeverzeichnisse *pi* sowie allenfalls alle anderen Homeverzeichnisse auf dem Server unter deren Namen freigegeben – vorausgesetzt, für die weiteren User besteht auch sowohl eine Linux-User-ID als auch ein per *smbpasswd* erstellter Samba-User samt Passwort!

Obige Angabe von '***valid users = %S'*** gewährleistet dabei, dass jeder Samba-User nur jeweils auf sein *eigenes* Home-Verzeichnis, d.h. auf seine *eigene* Home-Freigabe zugreifen kann (und nicht auf andere).

Eine Angabe von '*path* = ' ist nicht nötig, denn der Pfad zu den Homeverzeichnissen ermittelt der *smbd* Daemon aus der Datei */etc/passwd* . Vergewissern Sie sich davon mit: '***cat /etc/passwd***'

Aufgrund von '***read only = no'*** (oder alternativ per '*writable = yes'*) ist auch ein Schreibzugriff möglich!

Die Option '***browseable = no'*** dient (ein wenig) der Sicherheit: dadurch werden die „homes“-Freigaben versteckt und sind auf Ebene *server* nicht gleich für alle sichtbar!

Erstellen eines User-Accounts ohne Login-Rechte

Oft sollen Samba-User zwar Zugriffsrechte auf Samba-Freigaben haben, jedoch **kein Linux-Login-Recht** auf dem Server selbst. Für derartige User sind nebst Samba-Useraccounts und zugehörige Samba-Passworte trotzdem Linux-Useraccounts nötig, **jedoch mit gesperrtem Linux-Passwort** sowie einer Shell-Angabe von *'/bin/false'* in */etc/passwd,* wodurch sich der User eben *nicht* direkt am System anmelden kann (auch nicht via *ssh*).
Das Erstellen derartiger Useraccounts geschieht gemäss den man-Pages *useradd* und *smbpasswd* per:

```
$ sudo  useradd -m -s /bin/false  username     Erstellt Linux-User inkl. Homeverzeichnis und ohne Login-Shell
$ sudo  smbpasswd -a username                  Erstellt den Samba-User (mit interaktiver Passwortangabe)
```

- Erstellen Sie auf diese Weise einen User „**testuser**“ und testen Sie...

8 Der Member Server muss aber wie ein normaler Windows Domain Client zuerst in die *Active Directory Domain* aufgenommen werden, wozu Domänen-Administrationsrechte benötigt würden...

Ein Windows Client lässt aber gleichzeitig nicht mehrere User zu einem SMB-Server zu. Beim Testen mit mehreren User-Accounts ab dem gleichen Windows-Client tritt folglich ein Problem auf, weil der Windows-Client geöffnete Freigaben selbst dann offen hält, wenn das Explorer-Fenster geschlossen wird.

– Überprüfen Sie dies in einer Window-Commando-Console (nach *Start > Ausführen:* **cmd**) mittels:

> **net use**

Somit kann kein *anderer* Username angegeben werden solange man sich nicht bei Windows abmeldet - daran ändert auch ein Neustart des Samba-Daemons nichts!
Statt sich aber jedesmal mühsam erst ab- und anzumeden, kann die Verbindung zur Freigabe auch wie folgt in einer Windows *cmd*-Shell getrennt werden:

> ***net use \\servername\freigabename /d***

– oder gleich alle vorhandenen Serververbindungen trennen: (**Achtung**: auch jene zum FHNW-AD-Server!)

> ***net use * /d***

Gruppenfreigaben

Generell sollte das Verzeichnis */tmp* nicht freigegeben werden, da dieses Verzeichnis einerseits beim Booten meist gelöscht wird, andererseits von einigen Linux-Applikationen für temporäre Dateien verwendet wird. Man richtet Freigaben "für alle" besser auf andere, dedizierte Freigabeverzeichnisse ein.

Wo solche Gruppen-Freigabeverzeichnisse erstellt werden sollen, ist Ansichtssache: einerseits empfiehlt der aktuelle „*File Sytem Hierarchy Standard"* Server-Freigaben unter dem Verzeichnis */srv/* anzulegen, also z.B. unter */srv/samba/public,* andererseits spricht auch einiges dafür, derartige Gruppen-Freigabe-verzeichnisse unter */home/...,* z.B. auf */home/public* anzulegen – *ohne* zugehörigen User-Account!

- Der Ordner würde somit bei einem Backup aller Userdaten (unter */home/...*) ebenfalls gesichert.
- Oft wird das Stammverzeichnis */home/* auf eine separate Partition gemountet, womit diese (resp. die Userdaten) bei Neuinstallation des Betriebssystems erhalten bleiben kann (und damit auch das oben definierte Gruppenfreigabeverzeichnis). Auch bei einem Ausfall der Systempartition ist dies besser.
- Wenn tatsächlich ein Raspi "produktiv" als Samba-Server eingesetzt werden soll, sind Freigaben auf SD-Karten-Ordner jedoch langsam, voumenbegrenzt und fehleranfällig. Diesbezüglich besser sind Samba-Freigaben, welche auf eine externe, via USB3 angeschlossene HD oder SSD laufen.
  Beim *Raspberry Pi OS Lite* werden jedoch nicht wie bei der Desktop Version externe USB-Datenträger automatisch gemoutet, weshalb man die externen Diskpartitionen entweder nach jedem Booten wieder manuell mounten muss, oder man trägt diese Partitionen im richtigen Format in */etc/fstab* ein, sodass diese beim Booten automatisch gemountet werden. (Als Mountpoint könnte man einen Ordner unter */mnt/...* angeben oder gleich als */home* mounten, sodass die externen Homeverzeichnisse aufgrund der Samba *[homes]* Definition automatisch freigegeben würden). [9]

Ergänzen Sie folgenden Abschnitt in */etc/samba/smb.conf*

```
[public]
        path = /home/public
        read only = no
```

Natürlich muss das Freigabeverzeichnis */home/public* auch noch **erstellt** werden (per '*sudo **mkdir ...'**).*

Merke: Beim Zugriff auf die Freigabe wird der Netbios-User auf den gleichnamigen Linux-User gemapped! Ein Schreiben auf eine (beliebige) Freigabe erfordert also, dass der gemappte Linux-User auch Schreibrecht auf das Freigabeverzeichnis hat!

– Am einfachsten vergeben Sie also auf dem Freigabeverzeichnis „Schreibrecht für Alle" z.B. mit:

$ sudo **chmod 777 /home/public**

Erklärung: Die Angabe '*777*' oder '*a=rwx*' resp. '*ugo=rwx*' steht bekanntlich für **u**ser=**rwx**, **g**roup=**rwx** und **o**thers=**rwx** , d.h. sowohl der *Eigentümer* dieses Verzeichnisses wie auch Mitglieder der auf dem Verzeichnis angegebenen *Gruppe* wie auch *alle Anderen* User (*others*) dürfen den Verzeichnisinhalt lesen, schreiben (oder löschen) sowie das Verzeichnis „betreten".

9 Übrigens ist erst das Raspberry Pi 4 bezüglich Durchsatz von/zu USB-Disks einigermassen performant, denn erst bei diesem ist der USB3 Anschluss direkt am PCIe Bus angeschlossen. Alternativ könnte man auch ein Linux-Board mit SATA-Interface verwenden, wie das *BananaPi Pro,* und daran eine SATA-Harddisk oder -SSD anschliessen.

Da das Verzeichnis über *sudo* erstellt wurde, wird der Verzeichnis-Eigentümer wie auch die Gruppe auf '*root'* gesetzt.

Nun können alle Samba-User in die Freigabe *public* schreiben – vorausgesetzt es existieren für sie sowohl ein Samba- als auch ein Linux-User-Account!

Schreibt *UserA* nun ein File über die Freigabe, kann es von *UserB* zwar gelesen, jedoch nicht verändert werden, denn Samba resp. das Linux-Filesystem definiert standardmässig auf dem File den Ersteller-User als Eigentümer (also *UserA)* sowie die File-Permissions 644 (d.h ***u**ser=rw- **g**roup=r--* und ***o**thers=r--*) weswegen nur der File-Eigentümer *Schreibrecht* auf das File hat. Löschen könnten es die andere User jedoch, da das *Verzeichnis* ja *für alle* beschreibbar ist. Ebenso verhält es sich, wenn ein User ein Unterverzeichnis erstellt!
Standardmässig werden in Linux also die Dateirechte (d.h. *Owner, Group* und *Permissions*) **nicht „vererbt“!**[10]

Variante: Gruppenfreigabe ohne gemeinsame Schreibrechte:

Die User können also in entsprechend erstellte Freigaben nur die *eigenen* Files verändern und auch nur direkt im freigegebenen Ordern („*public*“) sowie in *eigenen Unterverzeichnisse darunter* schreiben - dies könnte natürlich gewollt sein:
Jeder User kann *sein* Unterverzeichnis erstellen und Files *darin* ablegen, welche anderen Usern zum Lesen zugänglich sind. Eine Unschönheit bliebt trotzdem: da das Verzeichnis *public* selbst ja für alle User beschreibbar ist, können andere User die Dateien, welche direkt unter *public* sind löschen!

Variante: Gruppenfreigabe mit gegenseitigem Änderungsrecht:

Sollte auch ein Ändern von Files aller *anderen* User möglich sein, muss Samba angewiesen werden, dass auf der betreffenden Freigabe die ***File-Permissions für neu erstellte Files*** auf ***g**roup=**rw-*** resp. ***o**thers=**rw-*** sowie bei neu erstellte Unterverzeichnisse auf ***g**roup=**rwx*** und ***o**thers=**rwx*** erstellt werden. Dies ist über zwei weitere Freigabe-Optionen einfach möglich:

```
[public]
        path = /home/public
        read only = no
        create mask = 0666
        directory mask = 777
```

Zugriffsbegrenzung auf User-Ebene:

Sollen ***nicht alle*** User sondern nur ***ausgewählte*** User Schreibrecht auf eine Gruppenfreigabe erhalten, könnte an stelle von '*read only = no*' wie folgt nur den ***namentlich erwähnten*** Usern Schreibberechtigung gegeben werden:

write list = UserA UserB ...

Mit der Option '*valid users*' könnte weiter auch eine Einschränkung der leseberechtigten User erfolgen:

valid users = *UserA UserB UserC UserD* ...

Sofern die Anzahl User gross ist, wird dies unpraktisch. In diesem Fall wäre es komfortabler, statt die User einzeln in *smb.conf* aufzuführen, eine entsprechende Linux-Gruppe zu erstellen, z.B. *pubwriter* für die Schreibberechtigten und *pubreader* für die Leseberechtigten. Die Erstellung der Gruppen und Aufnahme der User könnte z.b. wie folgt geschehen:

```
$ sudo groupadd pubwriter                  neue Gruppen erstellen...
$ sudo groupadd pubreader
$ sudo usermod -a -G pubwriter UserA       gewünschte User diesen Gruppen zufügen...
$ sudo usermod -a -G pubwriter UserB
$ sudo usermod -a -G pubreader UserC
$ sudo usermod -a -G pubreader UserD
```

Danach können diese Unix-Gruppen in *smb.conf* verwendet werden:

```
[public]
    write list = @pubwriter
```

10 Durch setzten des „Set-Group-Id“ Flags auf einem Verzeichnis kann der Gruppenname „nach unten“ vererbt werden. (s. weitere Fussnote)

valid users = @pubwriter @pubreader
:

Beachten Sie, dass die gesamte Rechte-Prüfung immer noch durch den Samba-Daemon geschieht, denn auf Filesystem-Ebene haben ja *alle Files dieser Freigabe imm noch Permission 666 resp. die Verzeichnisse 777*

Die Users aus *pubreaders* dürften sich also nicht direkt auf dem System oder per SSH einloggen können *sonst wäre die Sicherheit im Freigabeordner kompromitiert*! Einen Lösungsansatz für dieses Problem ist unter Fussnote [11].

Zugriff auf Interface- und IP-Ebene einschränken

Möchte man den Samba-Server auf Interface- oder IP-Ebene einschränken, sodass *smbd* die Freigaben nur auf bestimmten IP-Adressen oder -Adressbereiche zulässt, könnte dies unter Abschnitt *[global]* über die Einträge „***interfaces***“ sowie „***bind interfaces only*****“** geschehen, wie folgendes Beispiel zeigt:

[global]
:
interfaces = eth0 lo
bind interfaces only = yes

Damit würde der Samba-Dienst auf das Interface eth0 eingeschränkt und Zugriffe z.B. vom WLAN-Interface verhindern.[12]

Printer-Freigaben

Samba kann nicht nur Verzeichnisse freigeben, sondern auch die auf dem Serversystem installierten Printer-Queues. Dabei leitet Samba die Printjobs einfach weiter zum vorhandenen Linux Printer-Subsystem, wobei diverse unterstützt würden. Gängig auf aktuellen Linux (und Apple OS-X) Systemen ist das Printsystem **CUPS** (Common Unix Printing System).

Da Windows-Clients die Printjobs auf dem Client-Rechner selbst formatieren, müssen im Problemfall die auf dem Samba-Server definierter Drucker nochmals als Raw-Printer (d.h. ohne Formatierung auf dem Samba-Printserver) definiert und freigegeben werden!

Sogar die Windows-Druckertreiber könnten auf den Samba-Server hochgeladen werden, sodass ein Windows-Client diesen beim ersten Verbinden mit dem Drucker automatisch installiert (vgl. *smb.conf.ori*)

Alternativ kann ein Windows-Client aber auch **direkt** auf einen **CUPS Printerserver** drucken. Microsoft bezeichnet dieses Protokoll als *IPP, resp. IPPS (Internet Printing Protocol Subsystem)*. Damit CUPS lokale Printer auch extern freigibt, müsste in */etc/cups/cupsd.conf* noch *die* entsprechenden Zugriff gewährt werden...[13]

Fehlersuche

Zur Fehlersuche sind Logfiles dienlich. Je nach Samba-Konfiguration (in */etc/samba/smb.conf)* werden diese mehr oder weniger detailliert entweder ins „globale“ Logfile (auslesbar via Command 'journalctl') oder z.B. in separate Logfiles unter ***/var/log/samba/*** geschrieben. Betrachten Sie dieses Verzeichnis und schauen Sie sich die Dateien kurz an! (Z.B. mittels dem Pager 'less')

11 Ein korrektes Abbilden der Rechte auf Filesystem-Ebene ist wie folgt mittels „Set Group ID“ Flag möglich: Das Schreibrecht für *others* muss hierzu auf dem Freigabeverzeichnis entfernt werden ('*chmod 775 public')* sowie die Gruppe des Freigabeverzeichnisses geändert auf die Gruppe mit Schreibberechtigung ('*chgrp pubwriters public'*) sowie das „*Set Group ID“* Bits (*SGID*) auf diesem Freigabeverzeichnis durch '*chmod g+s public'* gesetzt werden. Dadurch wird der Gruppenname auf neu erstellten Files und Verzeichnissen vererbt! Zudem müsste in *smb.conf* auf der betreffenden Freigabe mittels '*create mask = 0664*' und '*directory mask = 2775*' das Schreibrecht und das „Set Group ID“ Flag für die Gruppe gesetzt sowie für *others* das Schreibrecht entzogen werden.
Sofern gleichzeitig mehrere User überschneidend sowohl Schreib- als auch das Leserecht in gleichen Verzeichnissen haben sollten, reicht dieser Ansatz jedoch nicht mehr. Dies wäre höchstens mittels „*Extended ACLs*“ zu lösen...

12 Das Loopback-Interface (lo) muss wie im Beispiel ebenfalls zugelassen werden! Alternativ resp. ergänzend zur Interface-Angabe könnte auch ein oder mehrere IP-Netzbereich angegeben werden (vgl. '*man smb.conf*' -> interfaces)

13 In *cupsd.conf* ändern von *'Listen localhost:631'* auf *'Listen 631'* sowie am Ende des Abschnitts *<Location />* zufügen z.B. von z.B. *Allow 192.168.0.0/24* um den Clients aus diesem IP-Bereich Zugriff zu gewähren.

Schlussbemerkungen

- Wird beim Zugriff auf einen Samba-Server auch Client-seitig ein Linux-System verwendet und die Freigaben auf dem Client-System direkt in den Verzeichnisbaum gemountet (per manuellem *mount* oder via */etc/fstab*), wendet Samba automatisch die **„*Samba Unix-Extensions*“** an.
  Diese *„Unix-Extension“* unterstützen auch die unter Linux oft verwendeten symbolischen Links (*Symlinks*) sowie die Linux *Permissions*. [14]

- An Stelle von Samba wurde früher im POSIX-kompatiblen Umfeld (und damit auch im Linux-Umfeld) vorwiegend **NFS** zur Dateifreigabe resp. als **„Remote-Filesystem“** verwendet.
  NFS ist in der Grundkonfiguration zwar sehr einfach zu konfigurieren und ein sehr effizientes Protokoll, beinhaltet jedoch in der Grundkonfiguration **keine Benutzer-Authentifikation**, weshalb **NFS** in der Grundkonfiguration **sicherheitstechnisch äusserst bedenklich** ist und **nur in isolierten sicheren Bereichen** (wie z.B. zuhause) verwendet werden sollte, also keinesfalls über's Internet oder an unsicheren LANs wie hier an einer Schule oder in einem Firmen-LAN!
  Nur unter Verwendung neuerer NFS-Versionen (wie **NFS4 mit Kerberos**) ist eine sichere Authentifikation via „*Kerberos*“ Protokoll gewährleistet. Eine derartige Installation ist aber bei den meisten Distributionen recht aufwändig – deutlich aufwändiger als eine sichere Samba-Konfiguration.
- Will man eine Dateifreigabe aus resp. über das Internet zugänglich machen, sollte man aus Sicherheitsgründen weder Samba- und schon gar nicht NFS direkt "nach Aussen" zugänglich machen!
- Soll es wirklich SMB oder NFS über's Internet zugänglich sein, verwendet man am Besten einen VPN Tunnel. Bei der FHNW sind die Dateifreigabenvon Extern ja auch nur via VPN zugänglich. Die Meisten NAT-Router haben eine VPN-Router Funktionalität, welche sich einfach aktivieren lässt.
- Die andere Variante ist SSH welches die gleiche Sicherheit bietet wie ein VPN: Ein SSH-Server bietet nebst einer sicheren Consoleverbindung ja bekanntlich auch die Möglichkeit, einzelne Dateien oder auch ganze Verzeichnisse einfach und sicher mittels ***'scp'*** oder ***'sftp'*** zwischen Linux- resp. Unix-ähnlichen Systemen transferiert werden.
  Dies klappt sogar ab der grafischen Oberfläche aus dem Linux Dateimanager *nautilus* (in der Form: sftp://user@host/) oder von und zu Windows-Clients unter Verwendung von „***WinScp***“.
  Um Angriffe auf den SSH-Port zu vermeiden, definiert man am Besten den SSH-Serverport von default 22 auf einen "krummen" Wert im Bereich zwischen 10'000 … 60'000 um.
- Alternativ eigenen sich für Dateifreigabe aus dem Internet natürlich auch Cloud-Lösungen wie *Dropbox* oder Microsofts *Sharepoint* etc. Eine eigene Cloudlösung werden Sie dann im nächsten Versuch kennen lernen...

14 ... denn an Windows-Clients wird über Samba bei einem Zugriff auf einen Symlink das referenzierte Objekt geliefert – also die Datei – denn das Windows-Dateisystem kennen ja die Symlink-Funktionalität nicht.
Bei Anwendung der „*Samba-Unix-Extension*“ wird auch die „*effektive UID*“ bei den Owner- und Group-Permissions als Fileattribute verwendet. Als Konsequenz müssen gleichnamige User auf beiden Seiten die gleiche UID-Nummer haben (vgl. Command '*id*'), wobei Ubuntu ja dem ersten User jeweils die UID 1000 zuweist!
Falls diese nicht übereinstimmt, müssen entweder die „*Unix-Extensions*“ ausgeschaltet werden – entweder clientseitig per entsprechende Mount-Option oder serverseitig im File *smb.conf* oder eben alternativ die UID-Nummern der User auf der einen oder anderen Seite mittels Commando '*usermod*' entsprechend geändert werden, sowie der „*Owner*“ auf allen Files nachgeführt werden (per '*chown -R ...*').