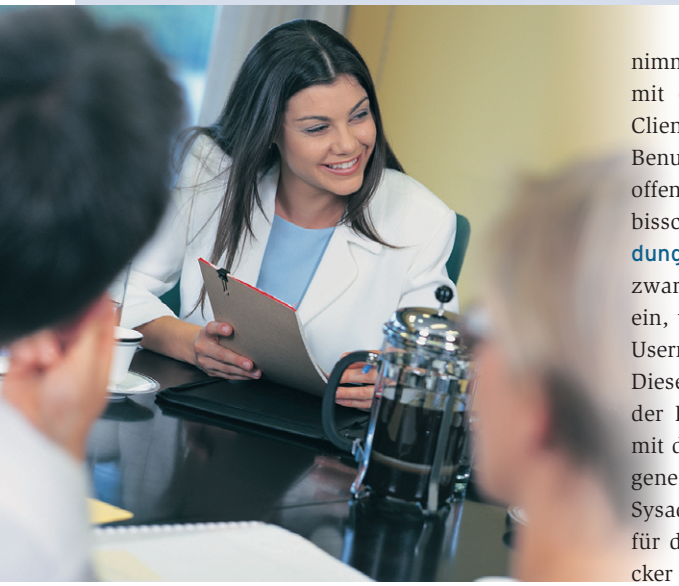
Linux-Client druckt im Windows-Netzwerk

Eindruck durch klasse Referenzen

Linux-Printserver bereiten ihren Benutzern kaum Probleme: LPR können alle Unix-Derivate, Windows-Clients drucken auf einen Samba-Server genauso wie auf Windows NT und 200X. Der umgekehrte Fall, ein Linux-Client druckt auf Samba- oder Windows-Servern, ist simpel. Es sei denn, auf dem Client arbeiten mehrere User. Harald Milz



Auf Client-Rechnern läuft immer häufiger Linux – eine begrüßenswerte Entwicklung. Darum schmerzt es kaum, dass in diesem Zuge nicht über Nacht alle Windows-Rechner und mit ihnen alle Microsoft-nahem Protokolle aus dem Unternehmensnetz verschwinden. Die Welt ist bekanntlich heterogen.

Wer nun mit seinem Linux-Client (hier: Suse Linux) die Dienste eines Windows- oder Samba-Printserver in Anspruch nimmt, hat dann kein Problem, wenn er mit einer einzigen Kennung auf dem Client auskommt. Wenn aber mehrere Benutzer von Suse aus drucken wollen, offenbart sich, dass Yast hier ein kleines bisschen unflexibel ist. Wie in Abbildung 1 zu erkennen ist, richtet das Tool zwar einen Cups-SMB-Drucker passend ein, verlangt aber gleich nach Domain, Usernamen und Passwort.

Dieses Account-Set gilt systemweit – jeder Benutzer auf dem Rechner druckt mit der einen User-ID des dort eingetragenen Serverzugangs. Zum Ärger jedes Sysadmin: So dürfen alle Benutzer die für den einen Server geschalteten Drucker benutzen. Auf der anderen Seite erreicht ein anderer Benutzer seinen eigenen Drucker vielleicht nicht, weil der mit dem falschen Passwort eingetragen ist. Überdies kommen die Ausdrucke aller Nutzer dieses Linux-Clients mit derselben Userangabe aus dem Drucker.

Trickreich

Mit ein bisschen Logik findet man den richtigen Lösungsansatz: Jeder Druckvorgang muss selbstverständlich mit den Credentials (Referenzen) des eingeloggten Benutzers erfolgen. Die hierfür nötigen Skripte wurden vom Autor dieses Artikels zwar auf Suse Linux 8.2 entwickelt, dürften aber in ähnlicher Form auch auf vielen anderen Linux-Varianten laufen.

Ein Seitenblick: Das Samba-Tool »smbclient« hilft beim Durchreichen der Credentials. Mit der Option »-A *Auth-File*« kann es Benutzerdaten aus einer Datei an sich ziehen. Sie könnte »$HOME/credentials« heißen und so aussehen:

```
username = HMILZ
password = XXXXXXX
domain = GECITS-EU
```

Es wäre sehr praktisch, diese Daten zugleich zur Authentifizierung gegenüber Druckerqueues einzusetzen, was eigentlich nicht vorgesehen ist.

Platzhalter

Wenn man auf eine Windows-Queue druckt, ruft Cups im Hintergrund das Druckprogramm »smbspool« des Samba-Client-Pakets auf. Die Koordinaten des Druckers liegen in der Datei »/etc/cups/printers.conf«. Listing 1 zeigt eine bereits angepasste Version. Der Parameter »DeviceURI« enthält für Domain, Usernamen und Passwort nur Platzhalter, die nachher ein Skript ersetzen wird. Genau diese Angaben sind nötig für die Personalisierung des Druckvorgangs.

Listing 1: »/etc/cups/printers.conf«

```
01 ...
02 <DefaultPrinter hplj4050>
03 Info HP Laserjet 4050
04 Location Unbekannter Drucker
05 DeviceURI smb://DOMAIN\USER:PASSWORT@demucms01/
   2OGN_HPLJ4050
06 State Idle
07 Accepting Yes
08 JobSheets none none
09 QuotaPeriod 0
10 PageLimit 0
11 KLimit 0
12 </Printer>
13 ...
```

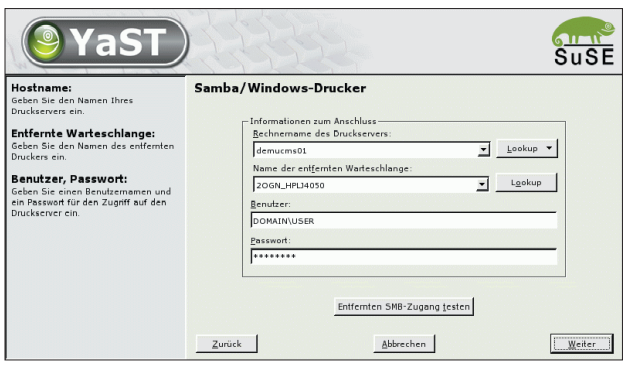


Abbildung 1: Yast erlaubt die Einrichtung eines Cups-Druckers auch für Windows-Warteschlangen – aber nur mit fest verdrahteten Benutzerdaten.

Das »smbspool«-Programm weiß allerdings nicht, welcher Benutzer gerade druckt, da Cups es leider mit Root-Rechten aufruft. Macht aber weiter nichts, Cups übergibt »smbspool« alle nötigen Angaben auf der Kommandozeile – insbesondere den Namen des Benutzers – und den »DeviceURI« als Umgebungsparameter:

```
smbspool [job] [user] [title] [copies]➥
 [options] [filename]
```

Damit ist der Hack nun komplett und das Shellskript »/usr/local/bin/smbspool.sh« (siehe Listing 2) kann die Aufgabe von »smbspool« übernehmen. (Nicht vergessen: »smbspool.sh« muss ausführbar sein!) Es holt die Benutzerdaten aus der Datei »$HOME/credentials« und baut den »DeviceURI« entsprechend um. Anschließend ruft es das originale »smbspool« auf.

Damit die Sache funktioniert, muss man nur »/usr/bin/smbspool« zu »/usr/bin/smbspool.ORIG« umbenennen und einen Link von dem Skript in Listing 2 zu »/usr/bin/smbspool« legen:

```
ln -s /usr/local/bin/smbspool.sh➥
 /usr/bin/smbspool
```

Vertraulichkeiten

Das Skript protokolliert seine Aktivitäten in die Datei »/tmp/smbspool.out«. Wer das nicht wünscht, kommentiert einfach die Zeilen 3 bis 5 aus. Es klang schon an: Das Skript läuft mit Root-Rechten – bei eigenen Erweiterungen sollte der geschätzte Anwender daher dem Thema Sicherheit genügend Aufmerksamkeit widmen. Zurück bleibt der Wunsch, Suse wäre selbst darauf gekommen, dass auf einem Linux-Client möglicherweise nicht nur ein einzelner Benutzer arbeitet. Dann ginge der angelegte symbolische Link auch nicht bei jedem Update des Smbclient-Pakets kaputt. *(jk)*

Der Autor

Harald Milz arbeitet als Senior-Systemberater bei der Firma CC Compunet AG & Co. oHG. Sein Themenschwerpunkt ist – wie könnte es denn anders sein – Linux.

Listing 2: »smbspool.sh«

```
01 #!/bin/sh
02
03 set -x
04 exec 1> /tmp/smbspool.out
05 exec 2> /tmp/smbspool.out
06
07 # smbspool {job} {user} {title} {copies} {options}
   [filename]
08
09 CREDS="/home/$2/credentials"
10
11 DOMAIN=$(sed -n 's/^domain = \(.*\)$/\1/p' $CREDS)
12 PASSWORD=$(sed -n 's/^password = \(.*\)$/\1/p' $CREDS)
13 USERNAME=$(sed -n 's/^username = \(.*\)$/\1/p' $CREDS)
14
15 DEVICE_URI=$(echo $DEVICE_URI | \
16              sed "s/DOMAIN/$DOMAIN/;
   s/PASSWORT/$PASSWORD/; s/USER/$USERNAME/")
17
18 export DEVICE_URI
19
20 exec /usr/bin/smbspool.ORIG "$@"
```