# Einbinden eines Linux Servers in eine Active Directory Domain (mit Samba – winbind)

11. März 2014

n Firmen werden meist Windows In Unternehmen findet man deshalb sehr oft eine Active Directory Domain. Was die Verwaltung der User und die Authentifizierung auf den unterschiedlichen Systemen enorm erleichtert. Das hinzufügen von Windowsrechnern in eine AD-Domain funktioniert ja relativ einfach und Problemlos. Etwas schwieriger wirds nur bei Unix/Linux Systemen da die Userverwaltung etwas anders funktioniert und das ganze richtig umgesetzt werden muss. Es gibt verschiedenste Arten wie man seinen Linux Server in die Domain bekommt, mit unterschiedlichen Vor- und Nachteilen. Vor allem was die Umsetzung der IDs der AD-Domain in UIDs und GIDs auf Unix Systemen betrifft hat man die Qual der Wahl. Die Testumgebung Eine Linux Kiste mit der aktuellen Debian Version (6) (diese läuft bei mir virtuell auf einem VMware ESX). Installiert ist eigentlich nichts. Nur Debian, Systemtools, und SSH Server wurden mit installiert. Also nix besonderes einfach nur ein standalone Linux Server der zu allem gemacht werden kann.

Im Unternehmen vorhanden sind mehrere Active Directory Domain's in eine davon kommt der neue Linux Server. Am Domain Controller läuft Windows Server 2003.

**Wissenswertes**

Eine Active Directory Domain besteht eigentlich aus 4 Komponenten, LDAP ein Verzechnisdienst, Kerberos ein auf Tickets basierendes Authentifizierungssystem (Single Sign On), CIFS ein Netzwerkfilesystem von Micrsoft (Windows Dateifreigabe) und DNS das klassische Domain Name System wie es auch im Internet verwendet wird. Also im Prinzip ist Active Directory ein Komplettpaket dieser Dienste. Die verwendeten Protkolle sind also nichts Unbekanntes oder Neues und werden auch in anderen Bereichen gerne verwendet.

Solltem einem diese Begriffe vollkommen neu sein (was wenn man eine AD-Domain verwaltet nicht der Fall sein sollte), kann man die entsprechenden Wikipedia Artikel lesen und sich auf der Technet Seite von Microsoft schlau machen.

**Die Netzwerkkonfiguration**

Da es sich hier um einen Server handelt bekommt der eine statische IP. Ein Server sollte ja immer mit der gleichen IP erreicht werden und ein entsprechender DNS Eintrag wird am Domain Controller angelegt. Einer
normalen Workstation mit Linux kann man alles per DHCP zuweisen (wichtig ist nur das die Domain, … korrekt vom DHCP verteilt wird). Eventuell werden auch Firewalls verwendet um Server abzusichern … ohne
statische IP's ist so was nicht so einfach möglich. Server -> statische IP.

Bei Debian befinden sich die Netzwerkeinstellungen in der Datei /etc/network/interfaces, die einfach mit einem Editor bearbeitet werden kann.

## Quellcode

1. /etc/network/interfaces
2. # This file describes the network interfaces available on your system

```
3. # and how to activate them. For more information, see interfaces(5).
4.
5. # The loopback network interface
6. auto lo
7. iface lo inet loopback
8.
9. # The primary network interface
10. allow-hotplug eth0
11. iface eth0 inet static
12. address 10.10.10.114
13. netmask 255.255.255.0
14. gateway 10.10.10.254
15. broadcast 10.10.10.255
```

Mein Testserver befindet sich im Netz 10.10.10/24 (das Subnet der Server im LAN)und hat die IP 10.10.10.114.

DNS Einstellungen werden wie nahezu bei jedem UNIX/Linux in die /etc/resolv.conf eingetragen. Als domain & search wird der FQDN der AD-Domain eingetragen. Als DNS-Server die Domaincontroller der Domain
(hier 4 Stück)

## Quellcode

```
1. /etc/resolv.conf
2. domain openitsolutions.local
3. search openitsolutions.local
4. nameserver 10.10.10.10
5. nameserver 10.10.10.20
6. nameserver 10.10.10.30
7. nameserver 10.10.10.40
```

Nun hat der Server seine IP und er kann die Namen der Domain auflösen. Damit ist der Server so weit fertig im Unternehmensnetzwerk und man kann schon das Netzwerk und das DNS-System im Unternehmen mit
dem Server verwenden. Allerdings der Server selbst ist noch unbekannt und AD-Domain Benutzer können noch nicht mit ihren Accounts auf den Server zugreifen.

**Installation der benötigten Pakete**
Damit der Linux Server auch mit den Domaincontrollern kommunizieren kann, muss dieser natürlich die verwendeten Protkolle verstehen.

- Kerberos
- CIFS
- DNS

DNS ist ja kein Problem da das auch unter Linux das am meisten genutzte Namenssystem ist. Damit der Server CIFS unterstützt wird SAMBA benötigt und Kerberos 5 damit der Server auch die Tickets der User
prüfen kann.

Folgende Pakete werden benötigt um den Server in die Domain zu integrieren:

- libkrb53
- krb5-admin-server
- krb5-kdc
- samba
- winbind
- ntpdate
- ntp

## Quellcode

1. root@schatzkiste ~ # agi libkrb53 krb5-admin-server krb5-kdc samba winbind ntpdate ntp

**agi? Funktioniert nicht?**

Tipp:

Nicht wundern wenn das Programm agi nicht gefunden werden kann. Ich verwende die zsh (eine ausgezeichnete Shell die mehr kann als die bash) mit der Konfiguration von GRML. GRML ist eine Linuxdistribution die auf Debian basiert und deren Entwickler diese Shell lieben. GRML verwendet in der Konfiguration der zsh Aliases für apt-get install, apt-get update, apt-get dist-upgrade, apt-cache search … (agi, au, adg, acs). Was einem einiges an Tipparbeit erspart.

>> **Zsh Installation & Konfiguration** <<

## Quellcode

1. root@schatzkiste ~ # apt-get install libkrb53 krb5-admin-server krb5-kdc samba winbind ntpdate ntp

## Quellcode

1. root@schatzkiste ~ # /etc/init.d/samba stop
2. Stopping Samba daemons: nmbd smbd.
3. root@schatzkiste ~ # /etc/init.d/winbind stop
4. Stopping the Winbind daemon: winbind.
5. root@schatzkiste ~ # /etc/init.d/ntp stop
6. Stopping NTP server: ntpd.
7. root@schatzkiste ~ #

**NTP**

Kerberos basiert ja wie schon erwähnt auf Tickets. So ein Ticket hat nur eine bestimmte Gültigkeitsdauer (so wie eine Tageskarte bei einem Musikfestival). Damit das Ticket des Users auch am Server funktioniert
darf es nicht abgelaufen sein. Es ist also sehr wichtig das der Linux Server die Uhrzeit genau sychron eingestellt hat wie der Domain Controller. Darum muss er sich die Zeit von der gleichen Quelle wie der/die Domain Controller holen. Oder wie meistens üblich, läuft am Domain Controller ein NTP-Server (Zeitserver) über den sich die an die Domain gebundenen

Systeme die Zeit holen.

Man muss also den vorhin installierten NTP-Client (das Paket ntp) so konifgurieren das dieser die zeit korrekt synchronisiert. Den ntp-Client kann man mit der Konfigurationsdatei /etc/ntp.conf konfigurieren. In
meinem Netzwerk läuft auf jedem Domaincontroller ein NTP-Server. Die Debian Server kommentiere ich aus und ersetze sie durch meine Domaincontroller.

## Quellcode

```
1.  /etc/init.d/ntp.conf
2.  # /etc/ntp.conf, configuration for ntpd; see ntp.conf(5) for help
3.
4.  driftfile /var/lib/ntp/ntp.drift
5.
6.
7.  # Enable this if you want statistics to be logged.
8.  #statsdir /var/log/ntpstats/
9.
10. statistics loopstats peerstats clockstats
11. filegen loopstats file loopstats type day enable
12. filegen peerstats file peerstats type day enable
13. filegen clockstats file clockstats type day enable
14.
15.
16. # You do need to talk to an NTP server or two (or three).
17. #server ntp.your-provider.example
18.
19. # pool.ntp.org maps to about 1000 low-stratum NTP servers. Your server will
20. # pick a different set every time it starts up. Please consider joining the
21. # pool: <http://www.pool.ntp.org/join.html>
22. #server 0.debian.pool.ntp.org iburst
23. #server 1.debian.pool.ntp.org iburst
24. #server 2.debian.pool.ntp.org iburst
25. #server 3.debian.pool.ntp.org iburst
26.
27. server ad01.openitsolutions.local
28. server ad02.openitsolutions.local
29. server ad03.openitsolutions.local
30. server ad04.openitsolutions.local
31.
32.
33. # Access control configuration; see /usr/share/doc/ntp-doc/html/accopt.html for
34. # details. The web page <http://support.ntp.org/bin/view/Support/AccessRestrictions>
35. # might also be helpful.
36. #
37. # Note that "restrict" applies to both servers and clients, so a configuration
38. # that might be intended to block requests from certain clients could also end
39. # up blocking replies from your own upstream servers.
40.
41. # By default, exchange time with everybody, but don't allow configuration.
```

```
42. restrict -4 default kod notrap nomodify nopeer noquery
43. restrict -6 default kod notrap nomodify nopeer noquery
44.
45. # Local users may interrogate the ntp server more closely.
46. restrict 127.0.0.1
47. restrict ::1
48.
49. # Clients from this (example!) subnet have unlimited access, but only if
50. # cryptographically authenticated.
51. #restrict 192.168.123.0 mask 255.255.255.0 notrust
52.
53.
54. # If you want to provide time to your local subnet, change the next line.
55. # (Again, the address is an example only.)
56. #broadcast 192.168.123.255
57.
58. # If you want to listen to time broadcasts on your local subnet, de-comment the
59. # next lines. Please do this only if you trust everybody on the network!
60. #disable auth
61. #broadcastclient
```

Danach kann man den ntp schon starten.

**Quellcode**

```
1. root@schatzkiste ~ # /etc/init.d/ntp start
```

MIt dme Befehl ntpq kann man den ntpd veranlassen eine Abfrage auf die Server zu machen um zu testen ob alle Server erreicht werden.

**Quellcode**

```
1. root@schatzkiste ~ # ntpq -p
2. remote refid st t when poll reach delay offset jitter
3. ==============================================================================
4. ad01.openitsolutions.local 192.168.3.1 4 u 7 64 1 0.435 7.169 0.000
5. ad02.openitsolutions.local 10.10.10.10 5 u 6 64 1 0.416 -91.311 0.000
6. ad03.openitsolutions.local 10.10.10.10 5 u 5 64 1 0.352 -19.135 0.000
7. ad04.openitsolutions.local 10.10.10.10 5 u 4 64 1 0.486 10.558 0.000
```

Damit hat man schon einen der wichtigen Schritte der Integration des Servers hinter sich.

**Kerberos**

Die Authentifizierung in der ActiveDirectory-Domain funktioniert mit Kerberos Tickets. Dazu muss Kerberos so konfiguriert werden das der Linux Server automatisch gegen die Domain authentifiziert. Das wird in
der /etc/krb5.conf definiert.

**Quellcode**

```
1. /etc/krb5.conf
```

```
2. [logging]
3. default = SYSLOG:INFO:DEAMON
4. kdc = SYSLOG:INFO:DEAMON
5. admin_server = SYSLOG:INFO:DEAMON
6.
7. [libdefaults]
8. default_realm = OPENITSOLUTIONS.LOCAL
9. krb4_config = /etc/krb.conf
10. krb4_realms = /etc/krb.realms
11. kdc_timesync = 1
12. ccache_type = 4
13. forwardable = true
14. proxiable = true
15.
16. [realms]
17. OPENITSOLUTIONS.LOCAL = {
18. kdc = ad01.openitsolutions.local
19. kdc = ad02.openitsolutions.local
20. kdc = ad03.openitsolutions.local
21. kdc = ad04.openitsolutions.local
22. admin_server = ad01.openitsolutions.local
23. default_domain = openitsolutions.local
24. }
25. [domain_realm]
26. .openitsolutions.local = OPENITSOLUTIONS.LOCAL
27. schatzkiste.openitsolutions.local = OPENITSOLUTIONS.LOCAL
28. .schatzkiste.openitsolutions.local = OPENITSOLUTIONS.LOCAL
```

Die Konfigurationsdatei ist in unterschiedliche Abschnitte unterteilt. [libdefaults], [relams], [domain_realm]… . Die genauen beschreibungen findet man in der Dokumentation von Kerberos (MIT)

[libdefaults]
Dieser Abschnitt enthält die Standardeinstellungen für alle Programme die Funktionen der Kerberos Lirary am System nutzen. In diesem fall Samba (winbind). Die wichtigste Anweisung ist hier eigentlich nur die
Definition der default_realm. Das ist die Standarddomain und hier wird einfach der FQDN der Active Directory – Domain eingesetzt.

[realms]
enthält die Parameter pro real. Hiersind die wichtigsten Parameter die Namen der DomainController (kdc) und den administrativen Server (admin_server) der Server der die Aufgaben wie Passwortwechsel …
übernimmt. Alle anderen Kerberos Server übernehmen die DAten von diesem Server.

In diesem Bereich muss natürlich die in der default_realm angegebene Domain genau definiert werden. Ist mehr als ein DomainController vorhanden müssen alle angegeben werden (ad01, ad02, ad03, ad04). Als

admin_server wird der erste Domain Controller defniniert (ad01). Der default_domain Eintrag ist nicht unbedingt notwendig aber sinnvoll damit auch Kerberos 4 genutzt werden kann.

[domain_realm]
Enthält die Informationen zur Umsetzung zwischen Domain-Namen (DNS) und den Kerberos-realm's. Hier werden einfach die beinhalteten Domains (Subdomains) der AD-Domain eingetragen.

Damit ist alles so eingestellt das Applikationen die Kerberos nutzen automatisch die Domain Controller nach der Gültigkeit der Tickets fragen.

Samba (Winbind)
Das Ziel ist den Server mit der Hilfe von Samba ein vollweriges mitglied der AD-Domain werden zu lassen. Also muss natürlich auch Samba so Konifguriert werden das es Kerberos zur Authentifizierung nutzt.
Samba wird mit der Datei /etc/samba/smb.conf konfiguriert und ist ähnlich wie die Konfiguration von Kerberos in verschiedene Bereiche unterteilt.

## Quellcode

```
1. /etc/samba/smb.conf
2. [global]
3. netbios name = SCHATZKISTE
4. workgroup = openitsolutions.local
5. realm = OPENITSOLUTIONS.LOCAL
6. password
7. server = ad01.openitsolutions.local, ad02.openitsolutions.local,
8. ad03.openitsolutions.local, ad04.openitsolutions.local
9. wins server = 10.10.10.10 10.10.10.20 10.10.10.30 10.10.10.40
10. #lokale Accounts auch verwenden
11. passdb backend = tdbsam
12.
13. security = ADS
14. encrypt passwords = true
15. log level = 0
16. idmap backend = idmap_rid:OPENITSOLUTIONS=10000-100000000
17. idmap uid = 10000-100000000
18. idmap gid = 10000-100000000
19. allow trusted domains = no
20. template shell = /bin/zsh
21. client use spnego = yes
22. client ntlmv2 auth = yes
23. winbind use default domain = yes
24. winbind enum users = yes
25. winbind enum groups = yes
26. winbind nested groups = yes
27. restrict anonymous = 2
28. domain master = no
29. local master = no
30. preferred master = no
31. os level = 0
```

```
32.
33. # log level = 10
34.
35. [homes]
36. comment = Home Directories
37. browseable = yes
38. read only = no
39. writeable = yes
40. create mask = 0700
41. directory mask = 0700
42. valid users = %S
43.
44.
45. ########################################
46. ### Testshares ##########################
47. #
48. # zum abschauen und als Beispiele für Berechtigungen
49.
50. #[public-readable]
51. # browseable = yes
52. # guest ok = yes
53. # comment = Everyone readable
54. # writeable = no
55. # path = /export/public
56.
57. #[domain-users]
58. # browseable = yes
59. # # Berechtigung nur für dir Gruppe Domain Users
60. # valid users = @"OPENITSOLUTIONS\Domain Users"
61. # path = /export/domain-users
62. # writeable = yes
63. # comment = Alle Domain User
64.
65. #[domain-admins]
66. # writeable = yes
67. # browseable = yes
68. # #Nur Domain Admins - Gruppe
69. # valid users = @"OPENITSOLUTIONS\Domain Admins"
70. # #hosts allow = 10.220.0.0/16
71. # path = /export/domain-admins
72. # comment = Alle Domain Admins
73.
74. #[local-users-adm-group]
75. # browseable = yes
76. # valid users = @adm
77. # path = /export/local-users
78. # writeable = yes
79. # comment = Lokale User in der adm-Gruppe
80.
```

```
#[domain-trabauer]
# browseable = yes
# valid users = OPENITSOLUTIONS\trabauer
# path = /export/flo
# writeable = yes
# comment = "OPENITSOLUTIONS\trabauer"

#[local-flo]
# browsable = yes
# valid users = flo
# path = /export/flo
# writeable = yes
# comment = Lokaler User flo

#[domain-local-users]
# browseable = yes
# path = /export/domain-local-users
# valid users = @"OPENITSOLUTIONS\Domain Users", @adm
# comment = Domain Users und lokale User aus der adm-Gruppe
        # read only = no

        Die [global]-Section
        enthält die allgemeinen Einstellungen von Samba und die default Werte für die anderen Bereiche. Die Konfiguration sieht komplizierter aus als sie ist

        netbios name = schatzkiste
```

Der NetBIOS Name des Servers
workgroup = openitsolutions.local

Der NetBIOS-Name der Active Directory Domain. Jede AD-Domain hat, eigentlich nur noch damit sie abwärtskompatibel ist, einen NetBIOS Namen.
realm = OPENITSOLUTIONS.LOCAL

Hier wird schlicht einfach der in der Kerberos-Konfiguration erstellte Realm eingetragen.
security = ADS

Defniert nur das Samba im Active Directory Services – Modul läuft.
allow trustet domains = no

Unbedingt auf no setzten wenn man als IDMAP-Backend den RID der Domain-User verwendet. Sonst wäre es ja möglich das zwei User aus unterschiedlichen Domains die gleichen Rechte auf der Linux Maschine
bekommen. Meist gibt es aber nur eine verwendete AD-Domain in einem Unternehmen, daher sollte es kein Problem darstellen vertrauten AD-Domains den Zugriff zum Server zu verweigern. Ansonsten muss man zu
anderen mühsamerem Möglichkeiten des idmappings zurückgreifen.
password server = ad01.openitsolutions.local,

ad02.openitsolutions.local, ad03.openitsolutions.local,
ad04.openitsolutions.local

Als Passwortserver werden ebenfalls die Domain Controller eingetragen. Dies ist notwendig für die Authentifizierung per NTLM.
wins server = 10.10.10.10 10.10.10.20 10.10.10.30 10.10.10.40

Die IP's oder Namen (DNS) der WINS Server im Netzwerk. Normalerweise läuft auf einem Domaincontroller auch ein WINS Server.

passdb backend = tdbsam

Die Art wie die Passwortinformationen von Samba gespeichert werden. Die ist der default Wert muss also nicht unbedingt angegeben werden.
encrypt passwords = true

Sollte klar sein
log level = 0

er WErt legt fest wie viel in den Logs mitgeschrieben wird. Bei Fehlersuche kann man den Wert auf 10 (Maximum) stellen und so alles genau mitverfolgen.
idmap backend = idmap_rid:OPENITSOLUTIONS=10000-100000000

IDMAP ist verantwortlich den Windows Usern und Gruppen eine eindeutige UserID und GroupID unter Linux zu geben. Diese ist entweder lokal was den großen Nachteil hat das die UIDs und GIDs nicht auf jedem Unix/Linux System im Unternehmen identisch sind. Oder man verwendet ein zentrales LDAP-Verzeichnis was allerdings wieder mehr Verwaltungsaufwand bedeutet. Eine sehr elegante und im ganzen Unternehmen konsistente Lösung ist, den RID (Ressource Identifier) des Windowsusers her zu nehmen und einen fixen Wert hinzuzählen. Der Nachteil ist natürlich das wenn mehrere Domains vorhanden sind RIDs doppeltvorkommen in beiden Domains. Daher muss trustet Domains der Zugriff auf den Samba-Server verweigert werden.
idmap uid = 10000-100000000

Der UID Bereich der AD-Domain User
idmap gid = 10000-100000000

Der GID Bereich der AD-Domain Gruppen
allow trusted domains = no

Vertrauten AD-Domains wird der Zugriff verweigert
template shell = /bin/zsh

Die Standardshell von Domain Usern
client use spnego = yes

Aktiviert die Kerberos Authentifizierung für Freigaben
client ntlmv2 auth = yes

Deaktiviert das unsicherere NTLMv1. Achtung NT4

winbind use default domain = yes

Ermöglicht den Login von Domain Usern ohne Angabe der Domain. Es wird automatisch angenommen das der User aus der defualt Domain kommt
winbind enum users = yes

Ermöglicht die Auflistung von Domainusern. Bei großen Domains sollte das wenn es zu Performanceproblemen kommt deaktivieren
winbind enum groups = yes

Das gleiche für Domain Gruppen.
winbind nested groups = yes

Man kann unter Windows Gruppen verschachteln und so Rechte vererben. Hier mit wird aktiviert das auch alle Rechte von verschachtelten Gruppen aktiv sind.
restrict anonymous = 2

Verhindert das nicht authentifizierte User Infos aus der AD-Domain abfragen können
domain master = no

Samba ist kein Domain Controller also aus
local master = no

Samba soll auch nicht bei NetBIOS rein pfuschen also auch die Local Master Browser unterstützung wird deaktiviert. Es bedeutet nicht das Samba der local Master Browser wird wenn aktiv aber Samba nimmt beim
Wahlverfahren teil wenn diese Option auf yes gesetzt wird
preferred master = no

Ob Samba eine Wahl zum Local Master Browser veranlassen soll oder nicht. In einem Netzwerk mit Active Directory und halbwegs aktuellen Windows Clients (XP und neuer) ist das klassische NetBIOS komplett überflüssig. Es verursacht nur unnötige Broadcasts …
os level = 0

Ist ein Wert der bei der Wahl zum local master browser eine Rolle spielt. Der Rechner/Server mit dem höchsten os level gewinnt im die Wahl. Ist das os level gleich dann entscheidet die uptime des Servers/Rechner. Der Gewinner übernimmt dann die Verwaltung der NetBIOS Namen. Normalerweise ist ein Wins Server vorhanden der die NetBIOS Namen verwaltetn

Damit ist Samba so weit konfiguriert und man kann seine Shares anlegen (Siehe Beispiele in meiner Konfiguration)

**Einbinden der Domain in den Server**
Nun kann zwar Samba die Domain nutzen der Server selbst ist aber immer noch ein standalone System. Damit man sich auch mit Domain Benutzern am Server anmelden kann muss winbind noch als Authentifizierungsmethode eingebunden werden.

**NSSWITCH**
Dem System muss auch mitgeteilt werden welche User am System vorhanden sind. Zusätzlich zu den gewohnten Files (/etc/passwd, /etc/groups) sind die User auch über winbind zu finden. Also wird die nsswitch um winbind ergänzt. Man kann auch optional die hosts um netbios erweitern. Ist aber da AD eingesetzt wird unnötig, weil am Domain-Controller auch ein DNS-Server läuft. Auch das Anlegen der Userverzeichnisse … soll automatisiert geschehen. Ein Admin ist schließlich faul (zumindest dämliche Routinearbeit versucht man zu meiden).

## Quellcode

```
/etc/nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd: compat winbind
group: compat winbind
shadow: compat

hosts: files dns
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: nis
```

Dem System sind jetzt also die User der Domain bekannt, nur wie die Arten der Authentifizierung müssen ja auch noch festgelegt werde um sich endgültig am Server anmelden zu können.

**PAM Pluggable Authentication Modules**
Das geschieht unter Debian mit den PAM (Pluggable Authentication Modules). PAM ist eine Entwicklung von Sun Microsystems und steht mittlerweile auch für Linux, AIX, HP-UX, FreeBSD, MacOSX zur Verfügung.
Auch solche Systeme lassen sich also im Prinzip auf diese weise an eine AD-Domäne binden. Für winbind gibt es ein PAM-Modul das nur eingebunden werden muss um es zur Authentifizierung am Server zu nutzen.

## Quellcode

```
/etc/pam.d/common-auth
#
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
```

```
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.). The default is to use the
# traditional Unix authentication mechanisms.
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
auth sufficient pam_winbind.so
auth [success=2 default=ignore] pam_unix.so nullok_secure
auth [success=1 default=ignore] pam_winbind.so krb5_auth krb5_ccache_type=FILE
    cached_login try_first_pass
# here's the fallback if no module succeeds
auth requisite pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
auth required pam_permit.so
# and here are more per-package modules (the "Additional" block)
# end of pam-auth-update config
```

Damit kann man sich theoretisch schon mit Domain Usern wenn der Server der Domain beigetreten ist. Nur wollen wir ja noch Homedirectories automatisiert anlegen lassen. Dafür gibt es ebenfalls ein PAM-Modul das einem die Arbeit abnimmt. Es legt die Homeverzeichnisse mit hilfe einer Vorlage an. Ich nehme als Vorlage den Systemstandard der bei den meisten LInux Distributionen unter /etc/skel zu finden ist. Beim anlegen eines Users mit useradd -m wird dieser Ordner kopiert. Also verwende ich auch einfach diese Vorlage und geben beim PAM Modul den Skel-Ordner als Option an. Die angegebene umask definiert welche Berechtigungen bei neu angelegten Homeverzeichnissen gesetzt werden.

## Quellcode

/etc/pam.d/common-session1

```
#
# /etc/pam.d/common-session - session-related modules common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define tasks to be performed
# at the start and end of sessions of *any* kind (both interactive and
# non-interactive).
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules. See
# pam-auth-update(8) for details.
```

```
16.
17. # here are the per-package modules (the "Primary" block)
18. session [default=1] pam_permit.so
19. # here's the fallback if no module succeeds
20. session requisite pam_deny.so
21. # prime the stack with a positive return value if there isn't one already;
22. # this avoids us returning an error just because nothing sets a success code
23. # since the modules above will each just jump around
24. session required pam_permit.so
25. # and here are more per-package modules (the "Additional" block)
26. session required pam_unix.so
27. session optional pam_winbind.so
28. # end of pam-auth-update config
29.
30. session required pam_mkhomedir.so skel=/etc/skel umask=0026
```

Gut die Homezerzeichnisse werden hiermit angelgt. Nur eine Kleinigkeit fehlt in /etc/skel. Es wird die zshrc als Standardshell verwendet und ich will meinen User schon eine brauchbare Konfiguration mitgeben.

## Quellcode

```
1. /root@schatzkiste ~ # cp /root/.zshrc /etc/skel
2. root@schatzkiste ~ # la /etc/skel
3. total 164
4. drwxr-xr-x 2 root root 4096 Aug 18 15:56 ./
5. drwxr-xr-x 72 root root 4096 Aug 17 12:54 ../
6. -rw-r--r-- 1 root root 220 Apr 10 2010 .bash_logout
7. -rw-r--r-- 1 root root 3184 Apr 10 2010 .bashrc
8. -rw-r--r-- 1 root root 675 Apr 10 2010 .profile
9. -rw-r----- 1 root root 143110 Aug 18 15:56 .zshrc
```

Nun ist die Konfigurations großteils abgeschlossen.

**Mit dem Server der Domain beitreten**

Die Konfiguration ist eigentlich fertig der Server muss nur noch der Domain beitreten. Der Domain Controller vertraut dem neuen Server ja noch nicht.

Samba und winbind müssen erst mal gestoppt werden.

## Quellcode

```
1. root@schatzkiste ~ # /etc/init.d/samba stop
2. Stopping Samba daemons: nmbd smbd.
3. root@schatzkiste ~ # /etc/init.d/winbind stop
4. Stopping the Winbind daemon: winbind.
5. root@schatzkiste ~ # ps -ef | grep winbind
6. root 2909 2835 0 12:07 pts/0 00:00:00 grep winbind
7. root@schatzkiste ~ # ps -ef | grep smbd
8. root 2912 2835 0 12:07 pts/0 00:00:00 grep smbd
```

Danach werden die Datenbanken von Samba gelöscht. Keine Angst die enhält noch keine wichtigen Daten.

**Quellcode**

```
1. root@schatzkiste ~ # rm -rf /var/lib/samba/*
2. zsh: sure you want to delete all the files in /var/lib/samba [yn]? y
```

Mit dem Tool net ist das beitreten einer Domain recht einfach

**Quellcode**

```
1. root@schatzkiste ~ # net ads join createcomputer="AD-Server/Linux-Unix" -U
   trabauer
2. Enter trabauer's password:
3. Using short domain name -- OPENITSOLUTIONS
4. Joined 'SCHATZKISTE' to realm 'openitsolutions.local'
```

Tritt der Domain bei und erstellt den Computeraccount in der Domain in der OU AD-Server/LInux/Unix. Mit -U muss ein User mit Domain Admin rechten angegeben werden. Nur diese sind berechtigt User und Computer an zu legen.

Nach dem Join sollte der Server in der Domain zu finden sein.

So nun ist es so weit Samba und Winbind können gestartet werden.

**Quellcode**

```
1. root@schatzkiste ~ # /etc/init.d/samba start
2. Starting Samba daemons: nmbd smbd.
3. root@schatzkiste ~ # /etc/init.d/winbind start
4. Starting the Winbind daemon: winbind.
```

**Letzte Tests**

Nun wird noch überprüft ob der Domain Controller dem Server vertraut. Ob die Userliste und Gruppenliste

**Quellcode**

```
1. root@schatzkiste ~ # wbinfo -t
2. checking the trust secret for domain OPENITSOLUTIONS via RPC calls succeeded
3. root@schatzkiste ~ # wbinfo -u
4. root@schatzkiste ~ # wbinfo -g
```

Man kann mit wbinfo -a auch eine Testauthentifizierung machen. Mehr Tests und kann man in der manpage von wbinfo nachlesen.

So nun der ultimative Test ein Login am SSH-Server mit Domain Account am Server.

**Quellcode**

```
1. trabauer@windose ~ $ ssh schatzkiste
2. trabauer@schatzkiste's password:
3. Creating directory '/home/OPENITSOLUTIONS/trabauer'.
```

4. Linux schatzkiste 2.6.32-5-amd64 #1 SMP Tue Jun 14 09:42:28 UTC 2011 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
trabauer@schatzkiste ~ % echo $HOME
/home/OPENITSOLUTIONS/trabauer
trabauer@schatzkiste ~ % la
total 204
drwxr-x--x 2 trabauer domain users 4096 Aug 18 16:31 ./
drwxr-xr-x 3 root root 4096 Aug 18 16:30 ../
-rw-r----- 1 trabauer domain users 220 Aug 18 16:30 .bash_logout
-rw-r----- 1 trabauer domain users 3184 Aug 18 16:30 .bashrc
-rw-r----- 1 trabauer domain users 675 Aug 18 16:30 .profile
-rw-r----- 1 trabauer domain users 34323 Aug 18 16:30 .zcompdump
-rw------- 1 trabauer domain users 44 Aug 18 16:31 .zsh_history
-rw-r----- 1 trabauer domain users 143110 Aug 18 16:30 .zshrc
trabauer@schatzkiste ~ %

Das wars, Gratulation der Debian Server ist hiermit brauchbar in eine Domain integriert! Das Beispiel sollte ine brauchbare Lösung für viele Unternehmen sein. Wie genau man den Server konfiguriert sollte man dich
etwas an die Gegebenheiten anpassen. Samba bietet noch sehr viele andere Möglichkeiten. Auch die Authentifizierung lässt sich ohne Samba realisieren allerdings bietet Samba schöne mechanismen zum mappen der
UIDs … was den Server ideal in die Domain integriert.

**sudo**
Damit jeder Admin kann mit der Hilfe von Sudo mit seinem User arbeiten kann, muss noch z.B. die “Domain Admins” Gruppe hinzugefügt werden. Unter Debian ist dazu der Ordner /etc/sudoers.d Vorhanden in dem
Konfigfiles abgelgt werden können.

Darin kann man einfach ein File mit den entsprechenden Sudoers-Regeln anlegen.

**Quellcode**

1. /etc/sudoers.d/domainadmins1
2. %domain\ admins ALL = (ALL) ALL

Die Rechte auf das File müssen natürlich auch angepasst werden

**Quellcode**

1. chmo 0440 /etc/sudoers.d/domainadmins

Schon haben Domain Admins vollen Zugriff um den Server zu verwalten.