!!! ACHTUNG - evtl. veraltet - ACHTUNG !!!

Diese Seite wurde zuletzt am 9. Juli 2014 um 10:35 Uhr geändert.

Intro

M$ Windoof 7 einfach so zu „imagen“ und auf mehr als einem Rechner „deployen“ ist nicht wirklich praktikabel (gleicher Hostname, gleiche SSID,…).

M$ Windoof 7 „einfach so“ vorher per sysprep „zurückzusetzen“ bringt einen, da man bei jedem Client das nervige Setup von Hand nochmal durchgehen muss, auch nicht viiieeel weiter voran.

Auch eine individuelle Antwortdatei (unattend.xml) bringt uns an dieser Stelle nicht einen Schritt näher ans Ziel, wir bewegen uns dabei lediglich auf einer Parallelstraße, da danach alle Rechner zwar unterschiedliche SSIDs aber gleiche Hostnamen, „product keys“,… haben. Wir müssen zwar nichts mehr eingaben, aber „falsche Antworten“… naja!

Mit Bordmitteln von M$ Windoof 7 wäre hier dann auch bereits schluss, weil das Betriebssystem zum letztmöglichen Zeitpunkt vor dem Deployment (z. B. eine Batchdatei, auf welche der Registry-Schlüssel „HKLM\SYSTEM\Setup\CmdLine“ verweist und welche ein kleines Wunder vollbringt, bevor windeploy.exe aufgerufen wird) noch keine Individualkennzeichen (Mac-Adresse oder Festplatten-ID) ermitteln kann. (Windows 7 steckt diesbezüglich halt immernoch in den Kinderschuhen!)

Wie gut nur, dass ich mit Clonezilla image und deploye! Das Netzwerk-Linux startet nach dem Zurückspielen und vor dem Starten des Systems einfach ein kleines aber feines Ruby-Skript (wie gut nur, dass ich es vorher nicht mit einem VB-Skript versucht habe!), welches die „gecachte“ unattend.xml ändert (was gemäß Micro$oft tunlichst vermieden werden soll, aber wunderbar funktioniert! → siehe "Search Order 3").

Okay, okay, nen bissl Shell-Skript und ne kleine SQLite-Datenbank sind auch noch mit von der Partie, aber das war es dann auch schon.

Warum ich das hier schreibe? Ganz einfach: Immer, wenn ich dachte, ich wäre fertig, hat mich Windoof eines besseren belehrt und wieder eine Unfähigkeit des Systems zwischen mich und mein Ziel geschissen (kein Schreibfehler!).

Nach der X-ten Neuinstallation und etlichen Runden „trail and error“ dachte ich mir, dass ich das Leid keinem anderen (außer meinen ärgsten Feinden wie Bill Gates, Steve Balmer,…) wünsche und das hier einfach mal niederschreibe.

So, genug Dampf abgelassen… hier die Fakten:

Festplatte wipen

Auf dem virtuellen Server, auf dem Clonezilla läuft, habe ich

- dc3dd (tarball) installiert
- in den PRERUN-Ordner (bei mir im Fall von „Clonezilla Box Mode“ liegt das in „/tftpboot/node_root/opt/drbl/share/ocs/prerun“) ein Skript mit folgenden Inhalt abgelegt:

```
dc3dd if=/dev/zero of=/dev/sda progress=on
```

(dc3dd, weil es gegenüber dcfldd syntax-kompatibel zu dd ist, aber zusätzlich ein „progressing“ bietet.)

M$ Windoof 7-Installation

Danach einfach

- Windoof installieren
- Administrator-Konto aktivieren (und optional mit einem Passwort versehen)
- in das Administrator-Konto umloggen
- alten Benutzer (inkl. Daten) löschen
- nach belieben mit Software versehen (z. B. WinFuture-UpdatePack, Firefox, FoxitReader, Microsoft Security Essentials,...)
- unattend.xml nach C:\Windows\Panther\unattend.xml kopieren (dort wird sie automatisch gefunden und sowieso gecached!)
- Mediaplayer-Netzwerkfreigabe-Dienst beenden (Eingabeaufforderung mit Administratorrechten öffnen):

```
net stop wmpnetworksvc
```

- hinterher dann sysprep anschieben:

```
C:\Windows\System32\sysprep\sysprep.exe /generalize /shutdown /oobe
```

(passt beides übrigens wunderbar in eine Batch-Datei, die man dann mit Administratorrechten ausführt)

Hier nun endlich das "Wunder"

Datenbank mit Client-Informationen

- sqlite3 installieren

```
aptitude install sqlite3
```

- /usr/local/lib/sysprep/sysprep_7.sqlite erstellen:

```
 cd /usr/local/lib/sysprep/
 sqlite3 sysprep_7.sqlite 'CREATE TABLE "clients" ("mac_address" VARCHAR(18)
PRIMARY KEY NOT NULL, "username" VARCHAR(20), "hostname" VARCHAR(15),
"productkey" VARCHAR(29));'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999aaaaaa','user-0100','dozent','AAAAA-AAAAA-AAAAA-AAAAA-
AAAAA');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
```

```
VALUES('001999bbbbbb','user-0101','client-0101','BBBBB-BBBBB-BBBBB-BBBBB-
BBBBB');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999cccccc','user-0102','client-0102','CCCCC-CCCCC-CCCCC-CCCCC-
CCCCC');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999dddddd','user-0103','client-0103','DDDDD-DDDDD-DDDDD-DDDDD-
DDDDD');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999eeeeee','user-0104','client-0104','EEEEE-EEEEE-EEEEE-EEEEE-
EEEEE');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999ffffff','user-0105','client-0105','FFFFF-FFFFF-FFFFF-FFFFF-
FFFFF');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999gggggg','user-0106','client-0106','GGGGG-GGGGG-GGGGG-GGGGG-
GGGGG');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999hhhhhh','user-0107','client-0107','HHHHH-HHHHH-HHHHH-HHHHH-
HHHHH');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999iiiiii','user-0108','client-0108','IIIII-IIIII-IIIII-IIIII-
IIIII');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999jjjjjj','user-0109','client-0109','JJJJJ-JJJJJ-JJJJJ-JJJJJ-
JJJJJ');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999kkkkkk','user-0110','client-0110','KKKKK-KKKKK-KKKKK-KKKKK-
KKKKK');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999llllll','user-0111','client-0111','LLLLL-LLLLL-LLLLL-LLLLL-
LLLLL');'
 sqlite3 sysprep_7.sqlite  'INSERT INTO "clients"
VALUES('001999mmmmmm','user-0112','client-0112','MMMMM-MMMMM-MMMMM-MMMMM-
MMMMM');'
```

(Okay, okay, diese Daten entsprechen nicht mal unseren... müsst Ihr also in jedem Fall anpassen. Man könnte das ganze auch in eine Datei schreiben und importieren oder mit ner GUI erledigen,... das überlass ich Euch!)

Ruby-Stuff

zurück auf den virtuellen Server, auf dem Clonezilla läuft:

- ruby installieren
- rubygems installieren
- gems installieren

```
 aptitude install ruby libsylite3-ruby
 cd tmp
```

```
wget http://production.cf.rubygems.org/rubygems/rubygems-1.3.7.tgz
tar xvfz rubygems-1.3.7.tgz
cd rubygems-1.3.7
ruby setup.rb
cd tmp
rm -rf /tmp/rubygems-1.3.7
ln -s /usr/bin/gem1.8 /usr/bin/gem
gem install activerecord macaddr
```

POSTRUN

Weil die POSTRUN-Skripte unter Clonezilla etwas umständlicher aktualisiert werden müssen, bedienen wir uns hier lediglich eines „Wrapper-Skripts“:

/tftpboot/node_root/opt/drbl/share/ocs/postrun/sysprep_wrapper:

```
#!/bin/bash
/usr/local/sbin/config_sysprep
exit 0
```

und PRE- & POSTRUN-Ordner "aktivieren"

Wollen wir nun das Skript im laufenden Betrieb (DRBL-Client läuft) ändern, können wir das auf dem DRBL-Server tun und es wirkt sich unverzüglich auch auf den Client aus (weil es in der NFS-Freigabe geändert wird).

/usr/local/sbin/config_sysprep:

```
#!/bin/bash
mkdir /mnt/sda2
ntfs-3g /dev/sda2 /mnt/sda2
/usr/local/sbin/sysprep_7.rb
rm /mnt/sda2/Windows/Panther/unattend.xml
mv /mnt/sda2/Windows/Panther/merged.xml
/mnt/sda2/Windows/Panther/unattend.xml
sync
umount /dev/sda2
exit 0
```

(Okay, okay, sauberer wäre es, alles auf eventluell auftretende Fehler zu überprüfen und abzufangen... aber so läuft es erstmal!)

Sysprep nachträglich "configurieren"

/usr/local/sbin/sysprep_7.rb erstellen:

```
#!/usr/bin/ruby
# needed gems: sqlite3-ruby, activerecord sowie macaddr
```

```
 require "rubygems"
 require "active_record"
 require "macaddr"
 require "rexml/document"
 include REXML
 ActiveRecord::Base.establish_connection( :adapter => "sqlite3", :database
=> '/usr/local/lib/sysprep/sysprep_7.sqlite' )
 class Client < ActiveRecord::Base
 end
 client = Client.find_by_mac_address( Mac.address.delete( ":-" ).downcase )
 input = File.open( '/mnt/sda2/Windows/Panther/unattend.xml', "r" )
 doc = Document.new( input )
doc.root.elements[["settings[@pass='oobeSystem']]/component[[@name='Microsof
t-Windows-Shell-Setup']]/UserAccounts/LocalAccounts/LocalAccount/Name"].text
= client.username
doc.root.elements[["settings[@pass='specialize']]/component[[@name='Microsof
t-Windows-Shell-Setup']]/ComputerName"].text = client.hostname
doc.root.elements[["settings[@pass='specialize']]/component[[@name='Microsof
t-Windows-Shell-Setup']]/AutoLogon/Username"].text = client.username
doc.root.elements[["settings[@pass='specialize']]/component[[@name='Microsof
t-Windows-Shell-Setup']]/ProductKey"].text = client.productkey
 output = File.new( '/mnt/sda2/Windows/Panther/merged.xml', "w" )
 output.puts doc
 output.close
 input.close
 exit 0
```

Verteilen und glücklich sein!

Wird nun das Image zurückgespielt, wird die Festplatte (in unserem Fall die erste, also /dev/sda)

1. gewiped
2. mit dem image „bedampft“
3. die „allgemeine“ unattend.xml „individualisiert“
4. das „Mini-Setup“ von M$ Windoof 7 OHNE lästige Fragen zu stellen durchlaufen
5. X-mal rebootet
6. der neu angelegte „normale“ Benutzer (trotz Passwort) automatisch angemeldet
7. gewartet, bis der Benutzer entnervt M$ Windoof 7 wiped und wieder Linux/BSD/Unix installiert! *harharhar*
8. FERTIG!