Security

- iptables Regeln entfernen und vollen Zugriff erlauben
- SELINUX
- Chroot-Jail für SCP/SFTP erstellen
- SuidBit auf Scripte funktioniert nicht
- SFTP Server mit Chroot und Usermapping (www-data)
- DDoS Angriffe abwehren

iptables Regeln entfernen und vollen Zugriff erlauben

Zum entfernen aller iptables-Regeln, löschen aller Chains und um wieder den vollen Zugriff auf das System zu erlauben sind folgende Zeilen nützlich:

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
```

SELINUX

- Ihr müsst nur erlauben, alles andere ist/bleibt verboten. Anstatt Dich Durch die Logs zu hangeln, machst Du einfach folgendes, während/nachdem Du einmal alle Anwendungen benutzt (hast):

```
audit2allow -M meine_policy -l -i /var/log/audit/audit.log
semodule -i meine_policy.pp
```

Chroot-Jail für SCP/SFTP erstellen

Diese Anleitung beschreibt die Einrichtung eines SCP-Userjails unter Linux. Getestet wurde diese Anleitung unter CentOS 5.2 (und 5.3 64bit) und einem Windowsclient mit WinSCP.

1. Zuerst müssen das EPEL-Repo und RPMFusion-Repo nach dieser Anleitung aktiviert werden.

2. Jetzt mittels „yum -y install scponly“ die alternative Shell für den SCP-Zugriff installieren.

3. Folgendes zusätzlich in /etc/shells eintragen:

```
/usr/bin/scponly
/usr/sbin/scponlyc
```

4. ein neues Verzeichnis /usr/local/setup_chroot anlegen und folgendes Script dort als create_chroot_user.sh ablegen. Das Orginal-Script aus dem scponly-RPM hat leider nicht fehlerfrei funktioniert. Ich habe das Script vereinfacht und bereinigt. Das Script nimmt das Password von der Kommandozeile.

/usr/local/setup_chroot/create_chroot_user.sh

```
#!/bin/sh

#

# 2009 Oliver Voelker <wiki(at)magenbrot.net>
#

# create an chroot-home

#


            # the following is a list of binaries that will be staged in the target dir
#BINARIES="/bin/sh /bin/bash /bin/cp /bin/ls /bin/mkdir /bin/mv /bin/pwd /bin/rm /bin/rmdi

BINARIES=
"/bin/ls /bin/mkdir /bin/mv /bin/pwd /bin/rm /bin/rmdir /usr/bin/id /usr/bin/sftp /usr/libe


# determine architecture

ARCH=`uname -i`


# a function to display a failure message and then exit

fail ( ) {

        echo -e $@

        exit 1

}


# "get with default" function

# this function prompts the user with a query and default reply

# it returns the user reply

getwd ( ) {

        query="$1"

        default="$2"
```

```
        echo -en "$query [$default]" | cat >&2
        read
 response
        if [ x$response = "x" ]; then
                response=$default
        fi
        echo $response
}

# "get yes no" function
# this function prompts the user with a query and will continue to do so
# until they reply with either "y" or "n"
getyn ( ) {
        query="$@"
        echo -en $query | cat >&2
        read
 response
        while [ x$response != "xy" -a x$response != "xn" ]; do
                echo -e "\n'y' or 'n' only please...\n" | cat >&2
                echo -en $query | cat >&2
                read
 response
        done
        echo $response
}

if [ x/usr/bin/ldd = x ]; then
        echo "this script requires the program ldd to determine which"
        fail "shared libraries to copy into your chrooted dir..."
fi
USE_PW=1
;
# we need to be root
if [ `id -u` != "0" ]; then
        fail "you must be root to run this script\n"
fi

if [ "x$2" = "x" ]; then
        fail "Usage: $0 <username> <password>"
fi

targetuser=$1
targetdir=/home/$targetuser

/usr/bin/install -c -d $targetdir
```

```
/usr/bin/install -c -d $targetdir/
dev
/usr/bin/install -c -d $targetdir/
usr
/usr/bin/install -c -d $targetdir/usr/
bin
/usr/bin/install -c -d $targetdir/usr/
sbin
/usr/bin/install -c -d $targetdir/usr/local

/usr/bin/install -c -d $targetdir/usr/local/
lib
/usr/bin/install -c -d $targetdir/usr/local/
bin
/usr/bin/install -c -d $targetdir/
lib
/usr/bin/install -c -d $targetdir/usr/
lib
/usr/bin/install -c -d $targetdir/usr/
libexec
/usr/bin/install -c -d $targetdir/usr/libexec/
openssh
/usr/bin/install -c -d $targetdir/
bin
/usr/bin/install -c -d $targetdir/
etc
if [ $ARCH = "x86_64" ]; then

  /usr/bin/install -c -d $targetdir/
lib64
  /usr/bin/install -c -d $targetdir/usr/
lib64
fi


for bin in $BINARIES; do

        /usr/bin/install -c $bin $targetdir$bin

done


LIB_LIST=`/usr/bin/ldd $BINARIES 2> /dev/null | /bin/cut -f2 -d\> | /bin/cut -f1 -d\( | /
bin/grep "^ " | /bin/sort -u`


LDSOFOUND=0

if [ -f /usr/libexec/ld.so ]; then

        LIB_LIST="$LIB_LIST /usr/libexec ld.so"

        LDSOFOUND=1

fi

if [ -f /lib/ld-linux.so.2 ]; then

        LIB_LIST="$LIB_LIST /lib/ld-linux.so.2"

        LDSOFOUND=1

fi


# 64bit
if [ -f /lib64/ld-linux.so.2 ]; then
        LIB_LIST="$LIB_LIST /lib64/ld-linux.so.2"
        LDSOFOUND=1
fi

if [ -f /usr/libexec/ld-elf.so.1 ]; then
        LIB_LIST="$LIB_LIST /usr/libexec/ld-elf.so.1"
        LDSOFOUND=1
fi

if [ $LDSOFOUND -eq 0 ]; then
        fail i cant find your equivalent of ld.so
fi
```

```
/bin/ls /lib/libnss_compat* 2>&1 > /dev/null
if [ $? -eq 0 ]; then
        LIB_LIST="$LIB_LIST /lib/libnss_compat* /lib/ld.so"
fi

# 64bit
/bin/ls /lib64/libnss_compat* 2>&1 > /dev/null
if [ $? -eq 0 ]; then
        LIB_LIST="$LIB_LIST /lib64/libnss_compat* /lib64/ld.so"
fi


if [ "x$LIB_LIST" != "x" ]; then
        for lib in $LIB_LIST; do
                /usr/bin/install -c $lib $targetdir/$lib
        done
fi
if [ $USE_PW -eq 0 ] ; then
    /usr/sbin/useradd -d "$targetdir//store" -s "/usr/sbin/scponlyc" $targetuser
    if [ $? -ne 0 ]; then
         fail "if this user exists, remove it and try again"
    fi
else
     useradd -n $targetuser -s "/usr/sbin/scponlyc" -d "$targetdir//store"
    if [ $? -ne 0 ]; then
         fail "if this user exists, remove it and try again"
    fi
fi

chown 0:0 $targetdir
if [ -d $targetdir/.ssh ]; then
        chown 0.0 $targetdir/.ssh
fi

if [ ! -d $targetdir//store ]; then
        echo -e "\ncreating  $targetdir/store directory for uploading files"
        /usr/bin/install -c -o $targetuser -d $targetdir/store
fi

# the following is VERY BSD centric
# i check for pwd_mkdb before trying to use it
if [ x = x ]; then
        /bin/grep $targetuser /etc/passwd > $targetdir/etc/passwd
else
        /bin/grep $targetuser /etc/master.passwd > $targetdir/etc/master.passwd -d "
$targetdir/etc" $targetdir/etc/master.passwd
        /bin/rm -rf $targetdir/etc/master.passwd $targetdir/etc/spwd.db
fi

rm -f $targetdir/usr/bin/groups
#gcc groups.c -o groups
cp /usr/local/setup_chroot/groups $targetdir/usr/bin/groups

cp /usr/local/setup_chroot/ld.so.conf $targetdir/etc
cp /lib/libnss_files.so.2 $targetdir/lib

mknod $targetdir/dev/null c 1 3
chmod 666 $targetdir/dev/null

targetuid=`id -u $targetuser`
targetgid=`id -g $targetuser`
cat /etc/passwd | awk -F":" '{if($3==0){print $0}}' > $targetdir/etc/passwd
#winscp seems to work bad with long names with "_" char - like "template_scp"
#so we cheats it by standard "user" name
dummyuser="user"
dummyhome="/store"
dummyshell="/usr/bin/oafish"
cat /etc/passwd | awk -F":" '{if($3=='$targetuid'){print "'$dummyuser
':"$2":"$3":"$4":"$5":'$dummyhome':'$dummyshell'"}}' >> $targetdir/etc/passwd

cat /etc/group | awk -F":" '{if($3==0){print $0}}' > $targetdir/etc/group
dummygroup="users"
cat /etc/group | awk -F":" '{if($3=='$targetgid'){print "'$dummygroup':"$2":"$3":"$4}}' >>
 $targetdir/etc/group

ldconfig
cp /etc/ld.so.cache $targetdir/etc/ld.so.cache
```

```
mkdir -p $targetdir/lib/tls/
cp /lib/libc.so.6 $targetdir/lib/libc.so.6

rm -rf $targetdir/store
mkdir $targetdir/store
chown -R $targetuser.users $targetdir/store

echo $2 | passwd --stdin $targetuser

if [ $ARCH = "x86_64" ]; then
  echo
"64bit OS, I will copy all libs (workaround) see http://www.magenbrot.net/wiki/linux/chroot

  rm -rf $targetdir/lib/* $targetdir/lib64/* $targetdir/usr/lib/* $targetdir/usr/lib64/*
  /bin/cp -L /lib/* $targetdir/lib/
  /bin/cp -L /lib64/* $targetdir/lib64/
  /bin/cp -L /usr/lib/* $targetdir/usr/lib/
  /bin/cp -L /usr/lib64/* $targetdir/usr/lib64/
fi
```

Falls jemand die vergebenen Passwörter nicht in der history finden will, einfach die letzte Zeile des Scripts durch folgende ersetzen: „passwd $targetuser“, dann wird das Kennwort beim Anlegen abgefragt. In Zeile 58 müsste dann natürlich noch die Abfrage auf $1 geändert werden.

5. Jetzt werden noch folgende Dateien in /usr/local/setup_chroot angelegt:

```
#include <stdio.h>

main()
{
        printf("root users\n");
        return 0;
}
```

```
/lib
/usr/lib
```

6. Die Datei groups.c muss noch kompiliert werden, der Gnu C-Compiler muss ggf. mit „yum -y install gcc“ nachinstalliert werden. Mittels „gcc groups.c -o groups“ wird das Binary gebaut. Diese Datei wird unter RedHat/CentOS und Konsorten gebraucht, um die vorhandenen Gruppen zu ermitteln. Da wir den Usern aber keinen Überblick über die wirklich vorhandenen Gruppen geben wollen, wird dieses Binary ins Chroot kopiert, das nur die Gruppen root und users ausgibt.

7. Jetzt kann mit folgendem Aufruf ein SCP-User erzeugt werden:

```
/usr/local/setup_chroot/create_chroot_user.sh <username> <passwort>
```

8. Mittels WinSCP läßt sich das Ganze dann testen. Das Userhome ist /home/<username>/store. Die User dürfen nicht aus /home/<username> heraus und haben außer auf /home/<username>/store keine Schreibrechte.

Debug-Modus bei Problemen

Mit folgendem Befehl läßt sich ein erweitertes Logging aktivieren. scponly logt nach /var/log/secure

```
echo 2 > /etc/scponly/debuglevel
```

und wieder deaktivieren:

```
echo 0 > /etc/scponly/debuglevel
```

Anmerkungen:

Bei 32bit Servern hat die oben gezeigte Anleitung immer problemlos funktioniert. Allerdings hatte ich bei 64bit Servern Probleme. Das Script bedarf hier noch einiger Arbeit. In /var/log/secure tauchten beim Verbindungsversuch folgende Meldungen auf:

```
May  6 13:36:21 files scponly[4846]: chrooted binary in place, will chroot()
May  6 13:36:21 files scponly[4846]: 3 arguments in total.
May  6 13:36:21 files scponly[4846]:    arg 0 is scponlyc
May  6 13:36:21 files scponly[4846]:    arg 1 is -c
May  6 13:36:21 files scponly[4846]:    arg 2 is /usr/libexec/openssh/sftp-server
May  6 13:36:21 files scponly[4846]: opened log at LOG_AUTHPRIV, opts 0x00000029
May  6 13:36:21 files scponly[4846]: determined USER is "testuser" from environment
May  6 13:36:21 files scponly[4846]: retrieved home directory of "/home/testuser//store" for use
May  6 13:36:21 files scponly[4846]: Setting homedir to /store
May  6 13:36:21 files scponly[4846]: chrooting to dir: "/home/testuser"
May  6 13:36:21 files scponly[4846]: chdiring to dir: "/store"
May  6 11:36:21 files scponly[4846]: setting uid to 500
May  6 11:36:21 files scponly[4846]: processing request: "/usr/libexec/openssh/sftp-server"
May  6 11:36:21 files scponly[4846]: Using getopt processing for cmd /usr/libexec/openssh/sftp-s
May  6 11:36:21 files scponly[4846]: running: /usr/libexec/openssh/sftp-server (username: testus
May  6 11:36:21 files scponly[4846]: about to exec "/usr/libexec/openssh/sftp-server" (username:
May  6 11:36:21 files scponly[4846]: failed: /usr/libexec/openssh/sftp-server with error No such
```

Das Binary und die via ldd angezeigten Libraries wurden aber installiert (bzw. durch das Script kopiert). Ich konnte das Problem dadurch lösen, das ich alle Libs in /lib, /lib64 und /usr/lib, /usr/lib64 in das Chroot kopiert habe. Ich habe das create_chroot_user.sh entsprechend angepasst (irgendwann muss ich das mal noch bereinigen), damit bei einem 64bit System alle Libs kopiert werden.

```
rm -rf /home/testuser/lib/* /home/testuser/lib64/* /home/testuser/usr/lib/* /home/testuser/usr/
lib64/*
/bin/cp -L /lib/* /home/testuser/lib/
/bin/cp -L /lib64/* /home/testuser/lib64/
/bin/cp -L /usr/lib/* /home/testuser/usr/lib/
/bin/cp -L /usr/lib64/* /home/testuser/usr/lib64/
```

SuidBit auf Scripte funktioniert nicht

Linux verhindert die Ausführung von Scripten mit gesetztem SuidBit, nur kompilierte Programme sind erlaubt. Wäre es anders könnte z.B. ein Script mit gesetztem SuidBit auf root durch den Besitzer ja nach belieben geändert und dann als root ausgeführt werden.

Soll nun trotzdem ein Script (Bash, Python, etc) als andere Benutzer ausgeführt werden, hilft z.B. folgendes kleine C-Programm:

```
#include <stddef.h>
#include <stdlib.h>
#include <unistd.h>
int main(void) {
  execl("/pfad/zum/script/script.sh", "argument1", "argument2", (const char *) NULL);
  return(EXIT_FAILURE);
}
```

Das Programm einfach mit „gcc -o script-exec script-exec.c“ kompilieren und mittels „chmod +s script-exec“ das SuidBit setzen. Mit chown noch dem passenden Besitzer einstellen.

SFTP Server mit Chroot und Usermapping (www-data)

Diese Anleitung wurde zuletzt mit Debian Stretch getestet. Der standard openssh-Dämon kann SFTP zwar schon länger. Chroot ist dabei ein ziemlich nervig einzurichten. Ein Mapping auf einen anderen Benutzer ist damit auch nicht möglich (das ist vor allem beim Zugriff auf /var/www nützlich, wo z.B. der SFTP-Benutzer ein anderer ist, als der mit dem der Webserver läuft).

Mit MySecureShell läßt sich das aber alles sehr bequem erledigen. Die Installation erfolgt einfach über: `apt-get install mysecureshell`

Danach wird die Konfigurationsdatei angepasst:

/etc/ssh/sftp_config

```
## MySecureShell Configuration File ##
# To get more informations on all possible options, please look at the doc:
# http://mysecureshell.readthedocs.org

#Default rules for everybody
<Default>
 GlobalDownload  0 #total speed download for all clients
     # o -> bytes   k -> kilo bytes   m -> mega bytes
 GlobalUpload  0 #total speed download for all clients (0 for unlimited)
 Download   0 #limit speed download for each connection
 Upload    0 #unlimit speed upload for each connection
 StayAtHome  true #limit client to his home
 VirtualChroot  true #fake a chroot to the home account
 LimitConnection  10 #max connection for the server sftp
 LimitConnectionByUser 5 #max connection for the account
 LimitConnectionByIP 5 #max connection by ip for the account
 Home   /home/$USER #overrite home of the user but if you want you can use
      # environment variable (ie: Home /home/$USER)
 IdleTimeOut  5m #(in second) deconnect client is idle too long time
 ResolveIP  true #resolve ip to dns
 IgnoreHidden  false #treat all hidden files as if they don't exist
 DirFakeUser  false #Hide real file/directory owner (just change displayed permissions)
 DirFakeGroup  false #Hide real file/directory group (just change displayed permissions)
 HideNoAccess  false #Hide file/directory which user has no access
 DefaultRights  0640 0750 #Set default rights for new file and new directory
 MinimumRights  0400 0700 #Set minimum rights for files and dirs
 ShowLinksAsLinks false #show links as their destinations
</Default>

<User sftp-www1>
 Home   /var/www
 ForceUser  www-data
 ForceGroup  www-data
</User>
```

Das mysecureshell Binary braucht noch das suid-Bit, um das Usermapping verwenden zu können: `chmod 4755 /usr/bin/mysecureshell`

Der User sftp-www1 wird dann wie folgt angelegt: `useradd -G www-data -s /usr/bin/mysecureshell sftp-www1`

Der User bekommt nun entweder einen SSH-Key in sein Home gelegt (/home/sftp-www, nicht /var/www) oder ein Passwort gesetzt und ist nach dem Einloggen in /var/www gebunden. Dateien, die der Benutzer hochlädt, werden dem Benutzer www-data zugeordnet.

DDoS Angriffe abwehren

Hier soll erklärt werden, wie DDoS (Distributed Denial of Service) Angriffe mit Linux Bordmitteln abgewehrt werden können. Je nach Angriff funktionieren diese Methoden allerdings nur schlecht oder gar nicht. In diesem Fall muss der Angriff soweit möglich auf vorgelagerter Hardware (Router) geblockt werden oder der Dienst hinter einen Anbieter wie CloudFlare umgezogen werden.

Wie erkenne ich einen DDoS?

Server hat hohe Load, Ping zeigt Packetloss, Login oder Arbeiten auf der Shell ist kaum möglich.

Abwehrmaßnahmen

Bei protocol-ddos TCP/UDP, Synflood

Syncookies auf den betroffenen Servern/Loadbalancern aktivieren:

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Angreifer identifizieren

Hosts mit den meisten Verbindungen:

```
netstat -atun | awk '{print $5}' | cut -d: -f1 | sed -e '/^$/d' |sort | uniq -c | sort -n
```

Anhand der Apache Zugriffslogs:

```
cat access.log | awk '{print $1}' | cut -d: -f1 | sed -e '/^$/d' |sort | uniq -c | sort -n | tai
```

Angreifer blocken

Per IP:

```
iptables -I INPUT -s <angreifer-IP> -j DROP
```

SYN-Pakete per iptables limitieren

```
/sbin/iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 4 -j DROP
/sbin/iptables -A INPUT -p tcp --syn -m connlimit --connlimit-above 25 -j REJECT --reject-with t
```