Gracefully Shut down non-standard virtuals

I like Xen. A lot. For me it is the best virtualization software out there. There is one pesky thing, however; if you don't have the right tools installed, Xen can not shut down your virtual (DOMU) nicely. Instead, it ends up doing a “destroy” on the domu, the same as pulling the power cord from your running workstation.

I call these my “pesky” domains, and they include Windows and the IPFire firewall. Yes, the pv drivers are available for Windows, but they seem a little flaky to me, so I don't always use them.

However, there is usually a way for the underlying DOM0 to shut down a running machine. With a unix style virtual, you could ssh to it and execute the halt command. With a windows virtual, you can use rpc to issue a remote shutdown command.

So, how to get it to work. Note: This is based on Debian and its successor in the server world, Devuan which I have grown to admire. I have not adapted it for FreeBSD.

My first attempt was to create a script that shut down the virtual first, then called shutdown (or reboot) for the DOM0. However, I wanted something more seamless, so I began investigating adding something to the init.d script for xen; /etc/init.d/xendomains (on Devuan).

Modify xen shutdown script

First, edit the script /etc/init.d/xendomains and find the line which reads do_stop_shutdown() and its associated curly brace under it. Modify it by adding the following (I have the two lines you put it after first, and the line which follows all in bold). **Only add the lines between the comments**.

```
do_stop_shutdown()
{
   # add following code
   if [ -x /opt/scripts/shutdown_nonstandard_domains ]
   then
      /opt/scripts/shutdown_nonstandard_domains
   fi
   # end of modification
   while read id name rest; do
```

This basically says *“if the file /opt/scripts/shutdown_nonstandard_domains exists and is executeable, run it”*.

Warning: you are messing with init.d scripts which are configured and installed by your package manager. This is fine, but just like modifying your car's computer, it will definitely void your warranty. Under Debian Wheezy, if the init.d script is to be changed during an upgrade, you'll receive a warning and need to manually edit the update.

Create script to manage shut down

Now, create the file /opt/scripts/shutdown_nonstandard_domains. Make it executable and make very sure it is set with permissions 700. This file may have sensitive information in it, so you should secure it. Also, I set the directory itself to the same so it can not be easily entered by non-root user.

shutdown_nonstandard_domains

```
mkdir -p /opt/scripts/
chmod 700 /opt/scripts
chown root:root /opt/scripts
echo '#!/bin/env bash' > /opt/scripts/shutdown_nonstandard_domains
echo "echo Shutting Down Nonstandard DOMUs" >>
/opt/scripts/shutdown_nonstandard_domains
chmod 700 /opt/scripts/shutdown_nonstandard_domains
chown root:root /opt/scripts/shutdown_nonstandard_domains
```

At this point, you have a null script that will be executed (but doesn't do anything except tell you that it is running. The sweet thing is, you can now run shutdown, halt, reboot, or even press the power button briefly and init.d will take care of safely shutting down your DOMU.

Add shutdown scripts as needed

I have some auxiliary scripts you can put in here. I'd suggest simply using shutdown_nonstandard_domains to call them, so, if you were using the IPFire version, you would modify your shutdown_nonstandard_domains to look like this.

```
#! /bin/env bash
echo Shutting Down Nonstandard DOMUs
/opt/scripts/ipfire_down
```

That way, you can test each script in turn for your machine, and add/remove whatever you need. For example, if you want to test the windows_down script, copy it to the /opt/scripts/ directory, set it up, then run it. See if your DOMU shuts down or not. THEN, you can add it to the shutdown_nonstandard_domains script.

Sample scripts

Following are a couple of sample scripts. I am definitely not the best bash coder, so laughing is not allowed. But, they work. They all have some serious security breaches, in that root/administrator usernames and passwords are stored clear text in the script itself.

IPFire Shutdown Script

Here is a sample script for IPFire.

ipfire_down

```
#! /bin/env bash

# safe shutdow of IPFire as a Xen DOMU
# this also works for opnSense/PFSense; just change 'halt' to
# 'poweroff'
# Author: R. W. Rodolico

# Copyright: 20151021 Daily Data, Inc.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see
&lt;http://www.gnu.org/licenses/>.

# script will call an IPFire installation and request it shut
# itself down. It will then wait until the router is shut down
# then terminate the DOM0 itself

# assumes root on DOM0 has ssh access to IPFire via public key
# and assumes private key has no password. To do this:
# ssh-keygen -t rsa -b 4096
# then, when it asks for a passphrase, just hit enter.
# copy /root/.ssh/id_rsa.pub to the IPFire installation as
/root/.ssh/authorized_keys
# as root from DOM0, then ssh to IPFire IP address and you should get
in with no passphrase.
# IPFire must be configured to allow ssh access via public key

# WARNING: this decreases security on your IPFire install. Anyone who
gains root access to your DOM0
#          has root access to your firewall. Protect your scripts and
at the first sign of a problem
#          kill your passphrase-less ssh access
# WARNING: I did not write a timeout for this script. It just checks
every 5 seconds to see if the
#          virtual shut down, from now until eternity.
```

```
# modify the following three variables for your installation
# must be the IP of your IPFire firewall
IPFIRE_IP=ip.of.router.here
# this must be the name as seen by your DOM0 of the IPFire firewall as
seen from xl list command
DOMU_NAME=ipfire
# the port your IPFire virtual listens on for ssh. 222 is the default
IPFIRE_PORT=222


# checks to see if IPFire still running using xl list and parsing it
for $DOMU_NAME
check_shutdown ()
{
   xl list | grep $DOMU_NAME > /dev/null || return 1
   return 0
}

echo "Shutting Down $DOMU_NAME"
# if the domain not running, simply exit
if check_shutdown
then
   # send halt command to virtual
   ssh -p $IPFIRE_PORT $IPFIRE_IP 'halt'
   # Check every 5 seconds to see if it has gone away
   while check_shutdown
   do
      echo -n '. '
      sleep 5
   done
fi
echo
echo "$DOMU_NAME Shut down"
```

As it says in the comments, you need to do some legwork. The basic idea for this is to create an ssh key without a passphrase, then tell IPFire to accept that coming from root. **This is bad stuff security wise** as if your DOM0 gets cracked, they have full root access to your router. So, secure the fire out of the DOM0.

The important part is the line which reads

```
ssh -p $IPFIRE_PORT $IPFIRE_IP 'halt'
```

The rest of it is just something that waits until the domain actually stops (possibly waiting forever, which is a deficiency in the script). Matter of fact, I may decide to rewrite the primary one a little differently, but not right now. That line ssh's to the IPFire installation and issues the halt command.

Windows Shutdown Script

To modify the script to shut down Windows machines, you add variables named username and PASSWORD, then change the ssh line above to use samba's net rpc command, as follows:

```
net rpc shutdown -f -I $IPADDRESS -U $USERNAME%$PASSWORD
```

windows_down

```
#! /bin/env bash

# remote shutdown of windows machine
# Author: R. W. Rodolico
#
# Copyright: 20151021 Daily Data, Inc.
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
# see
http://lifehacker.com/5275652/shut-down-your-windows-pc-remotely-from-l
inux
# Requires samba. Under Debian
# apt-get install samba-common
# see notes at bottom about how to set up the Windows machine

# modify the following for your installation
# ip of windows machine
IPADDRESS=ip.of.windows.server
# this user must have remote shutdown permission, generally a member of
the Adminmistrators group
USERNAME=username_of_admin_on_server
# the password for that user. This is a serious breach of security, so
think safe
PASSWORD=password_of_USERNAME
# this must be the name as seen by your DOM0 of the Windows machine as
seen from xl list command
DOMU_NAME=my-windows-server

# checks to see if virtual still running using xl list and parsing it
for $DOMU_NAME
```

```
check_shutdown ()
{
   xl list | grep $DOMU_NAME > /dev/null || return 1
   return 0
}

echo "Shutting Down $DOMU_NAME"
# if the domain not running, simply exit
if check_shutdown
then
   # send halt command to virtual
   net rpc shutdown -f -I $IPADDRESS -U $USERNAME%$PASSWORD
   # Check every 5 seconds to see if it has gone away
   while check_shutdown
   do
      echo -n '. '
      sleep 5
   done
fi
echo
echo "$DOMU_NAME Shut down"

# For Windows 7, it doesn't work because remote shutdown is disabled on
it. You will need to
# http://ubuntuforums.org/showthread.php?t=173440&page=7
#
http://www.howtogeek.com/howto/windows-vista/enable-mapping-to-hostname
c-share-on-windows-vista/
# A. shut down the firewall
# B. Add
#
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\S
ystem
# LocalAccountTokenFilterPolicy  DWORD  0x00000001 (1)
```

Again, see the attached script windows_down below for this. It has links to how I figured out how to do it and some caveats for Windows XP/Vista/7. Again, you have PASSWORD defined in the script, so anyone gaining access to this script then knows an administrators password on the server.