Setting up a Disk Wiping Operation with OpenBSD

Jim Brown, jpb@jimby.name, July 27, 2014

Introduction

Recently, I was faced with having to wipe a large number of disks that contained private information. The disks were mounted inside specialized hardware which made removing the disks somewhat difficult. The systems were older vintage x86 systems, circa 2000-2004 with some even older. Network interfaces were available, but there was no internal CD/DVD or floppy disk – only USB. And even though there were USB interfaces, the BIOS did not allow booting from USB.

In addition, the wiping had to be performed with a minimum of personnel, the inventory of all disks had to be strictly accounted for, and the evidence that wiping was thorough and complete had to be maintained. Oh- and did I mention that all of this had to cost the minimum possible?

Of course, Darik's famous Boot and Nuke (DBAN) came to mind, but in this case, it wasn't really usable unless the disks were removed from the system and wiped on a different machine. This was possible, but tedious. I started to contemplate a solution that I could use PXE and some simple scripting and realized that OpenBSD is ideal for this – PXE is well supported and the install program is a shell script.

I decided to try PXE booting, using the bsd.rd boot image and ksh scripting to see if I could find a way to wipe the disks without removing them from the system. I set up a PXE boot virtual server, and a set up another virtual image to boot from the network. PXE booting worked fine, and the required bsd.rd boot file was loaded and executed. I won't spend time on describing the DHCP/TFTP setup used as there are a number of good tutorials on setting these up for PXE booting. Check the man pages first.

Illustration 1 shows the OpenBSD bsd.rd install script with the "(I)nstall, (U)pgrade, (S)hell" option prompt. I selected (S)hell and used "dd if=/dev/zero of=/dev/rwd0c bs=1m" to wipe the drive. While this worked, I discovered that I didn't really have a way to show that the disk was, in fact, wiped with zeros. I'm technically inclined enough to know that it is wiped, but I can't show that fact to an auditor. The hexdump(1) utility is ideal for this, but it is not on the default ram disk on the boot media (Illustration 2). Some customization was necessary.

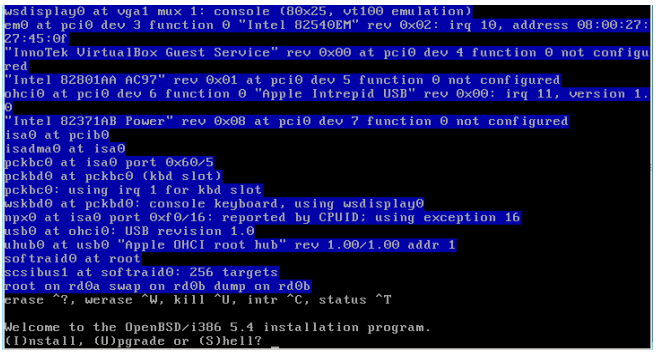

Illustration 1: OpenBSD Install Script and Disk Wiping via the Install Shell

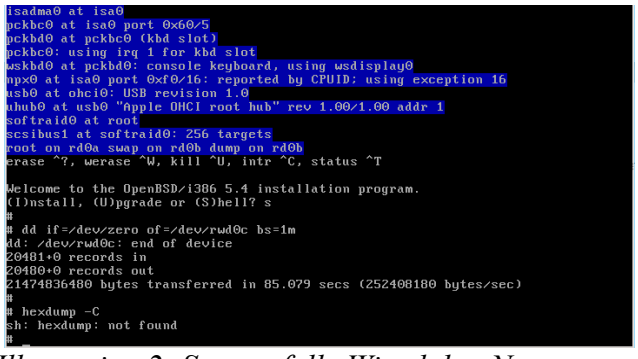

Illustration 2: Successfully Wiped, but No Verification Available

The source code for the install script is located in /usr/src/distrib/miniroot in the file dot.profile. This file gets renamed during the build process to .profile and it is called during the install. While it is possible to hack that file directly, a more flexible solution is to create a separate script and have it called by dot.profile. This solution is shown in Illustration 3.

```
        cat <<__EOT

Welcome to the $OBSD installation program.
__EOT
        while :; do
                read REPLY?'(I)nstall, (U)pgrade, (S)hell, or (W)(V)(R) Module:
'
                case $REPLY in
                i*|I*)  /install && break
                        ;;
                u*|U*)  /upgrade && break
                        ;;
                w*|W*|v*|V*|r*|R*) /wipeandverify && break
                        ;;
                s*|S*)  break
                        ;;
                !)      echo "Type 'exit' to return to install."
                        ksh
                        ;;
                !*)     eval "${REPLY#?}"
                        ;;
                esac
        done
fi
```

Illustration 3: Modified dot.profile Install Script Showing Wipe Module

```
pckbd0 at pckbc0 (kbd slot)
pckbc0: using irq 1 for kbd slot
wskbd0 at pckbd0: console keyboard, using wsdisplay0
npx0 at isa0 port 0xf0/16: reported by CPUID; using exception 16
usb0 at ohci0: USB revision 1.0
uhub0 at usb0 "Apple OHCI root hub" rev 1.00/1.00 addr 1
softraid0 at root
scsibus4 at softraid0: 256 targets
PXE boot MAC address 08:00:27:ee:02:82, interface em0
root on rd0a swap on rd0b dump on rd0b
erase ^?, werase ^W, kill ^U, intr ^C, status ^T

Welcome to the OpenBSD/i386 5.4 installation program.
(I)nstall, (U)pgrade, (S)hell, or (W)(V)(R) Module: v
Available drives:

wd0: 128-sector PIO, LBA, 8192MB, 16777216 sectors
sd0: 5120MB, 512 bytes/sector, 10485760 sectors
sd1: 5120MB, 512 bytes/sector, 10485760 sectors
sd2: 5120MB, 512 bytes/sector, 10485760 sectors

Enter drive you want to wipe/randomize/verify/ (wd0, sd1, etc): wd0
Ready to act on drive /dev/rwd0c

 (W)ipe drive, (R)andomize drive, (V)erify drive, (E)xit: v_
```

Illustration 4: Verifying a Wiped Drive

The drive wiping module is set up as a separate option on the dot.profile install prompt. Selecting that option runs the wipeandverify script as shown in Illustrations 4 and 5. Illustration 5 shows how the 'v' option was selected to actually verify the drive wiped earlier.

To verify a wiped drive, hexdump(1) was run with the '-C' option which shows the canonical output format. As noted in the manual page, "... any number of groups of output lines, which would be identical to the immediately preceding group of output lines (except for the input offsets), are replaced with a line comprised of a single asterisk ('*')", so the output of hexdump shows all zeros from the first byte of the drive to the last. In addition, the drive serial number is displayed. By copying, photographing, or printing this screen, you have proof enough to satisfy any auditor that a specific drive was completely overwritten with zeros.

```
Ready to act on drive /dev/rwd0c

 (W)ipe drive, (R)andomize drive, (V)erify drive, (E)xit: v
Verifying /dev/rwd0c ...

Try NOT to use Ctl-T to save real estate.
dd if=/dev/rwd0c bs=1m | hexdump -C
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
8192+0 records in
8192+0 records out
8589934592 bytes transferred in 79.353 secs (108248372 bytes/sec)
200000000
The following disk has been completely wiped:

Model: VBOX HARDDISK, Rev: 1.0, Serial #: VB6d595ded-7d5c50ee

Available drives:

wd0: 128-sector PIO, LBA, 8192MB, 16777216 sectors
sd0: 5120MB, 512 bytes/sector, 10485760 sectors
sd1: 5120MB, 512 bytes/sector, 10485760 sectors
sd2: 5120MB, 512 bytes/sector, 10485760 sectors

Enter drive you want to wipe/randomize/verify/ (wd0, sd1, etc): _
```

Illustration 5: Proof of Drive Wiping with hexdump(1)

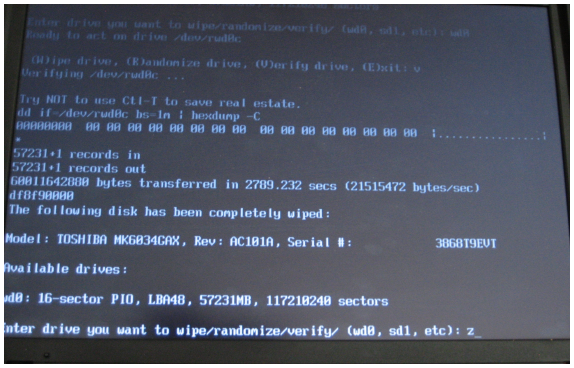

Illustration 6: Camera Image of a Wiped Drive

The previous illustrations have all been screen shots from a virtual machine. Illustration 6 is a camera photo from a real machine.

Customizations

To set all this up several customizations were necessary. The following lists of tasks will suffice to get you started on a similar setup:

- Download the OpenBSD source code and follow the directions for building a release as noted in Section 5 of the OpenBSD FAQ.
- Make a backup copy of the /usr/src/distrib/miniroot directory. Most of the action happens in this directory.
- Customize the file dot.profile to add the menu option as shown below:

```
        # Installing or upgrading?
        cat <<__EOT

Welcome to the $OBSD installation program.
__EOT
        while :; do
                read REPLY?'(I)nstall, (U)pgrade, (S)hell, or (W)(V)(R) Module: '
                case $REPLY in
                i*|I*)  /install && break
                        ;;
                u*|U*)  /upgrade && break
                        ;;
                w*|W*|v*|V*|r*|R*) /wipeandverify && break
                        ;;
                s*|S*)  break
                        ;;
                !)      echo "Type 'exit' to return to install."
                        ksh
                        ;;
                !*)     eval "${REPLY#?}"
                        ;;
                esac
        done
fi
```

- Create the file wipeandverify.sh as shown below. While the standard file copyright statement is not shown for brevity, you should include it in your file. This new script builds upon the work of Todd Miller, Theo de Raadt, Ken Westerback, Jason R. Thorpe, and possibly others. Remember that this is a Korn shell script, so include the she-bang line #!/bin/ksh at the top.

```
#!/bin/ksh
# [Standard copyrights here...]
# OpenBSD Wipe and Verify Script.  Copyright (c) 2014 by Jim Brown

(cd /dev/ && /bin/sh /dev/MAKEDEV random)

while :; do

        echo "Available drives:"
        echo
        dmesg | grep "^[ws]d[0-9][0-9]*:"
        echo
        read MYDEV?'Enter drive you want to wipe/randomize/verify/ (wd0, sd1, etc): '

        MYRAWDEV="/dev/r${MYDEV}c"
        echo "Ready to act on drive ${MYRAWDEV}"
        echo
```

```
        read REPLY?' (W)ipe drive, (R)andomize drive, (V)erify drive, (E)xit: '
        case $REPLY in
        w*|W*)  echo "Wiping $MYRAWDEV ..."
                echo "Use Ctl-T to view progress"
                echo "dd if=/dev/zero of=${MYRAWDEV} bs=1m"
                dd if=/dev/zero of=${MYRAWDEV} bs=1m
                ;;
        r*|R*)  echo "Randomizing $MYRAWDEV ..."
                echo "Use Ctl-T to view progress"
                echo "dd if=/dev/random of=${MYRAWDEV} bs=1m"
                dd if=/dev/random of=${MYRAWDEV} bs=1m
                ;;
        v*|V*)  echo "Verifying $MYRAWDEV ..."
                echo
                echo "Try NOT to use Ctl-T to save real estate."
                echo "dd if=${MYRAWDEV} bs=1m | hexdump -C"
                dd if=${MYRAWDEV} bs=1m | hexdump -C
                echo "The following disk has been completely wiped:"
                echo
                echo $MYRAWDEV | grep "/dev/rwd"  >  /dev/null
                if [ $? -eq 0 ] ; then
                   atactl $MYDEV identify | grep -i serial ;
                else
                   bioctl -q $MYDEV ;
                fi
                echo
                ;;
        e*|E*) echo "Exiting"
                break
                ;;
        esac

done
```

Sharp-eyed readers will have noticed that the script also has an '(R)(r)' option. The script also handles randomization – i.e. filling the disk with completely random data. It does this by using /dev/random instead of /dev/zero as the input source to the dd command. However, /dev/random is not installed by bsd.rd , so this script creates it as shown on the first line.

- There are a few more customizations necessary to get this script installed in the bsd.rd miniroot ram disk so it can be used. The machinery for building the miniroot ram disk is embedded in the release building scripts. To include this new script, hexdump, and atactl, modify three files that are used to specify the programs to be included in a release:

  ```
  /usr/src/distrib/i386/common/list
  /usr/src/distrib/miniroot/list
  /usr/src/distrib/ramdisk/list
  ```

  (Note that this is for the i386 architecture but in theory, it should work with any supported architecture.)

  Add the following lines in their respective sections.

  ```
  LINK    instbin                                 sbin/atactl

  LINK    instbin                                 usr/bin/hexdump

  SCRIPT  ${CURDIR}/../../miniroot/wipeandverify.sh       wipeandverify
  ```

and modify this existing line:

```
SPECIAL chmod 755 install upgrade wipeandverify
```

- Next, build another release. However this time, the build may fail on the final assembly of bsd.rd, so use the following lines to build the release

  ```
  # cd /usr/src/etc
  # make -k release
  ```

  The '-k' option will allow the make to finish if there are no significant errors on the primary target.

  If all goes well, the final output should look something like this:

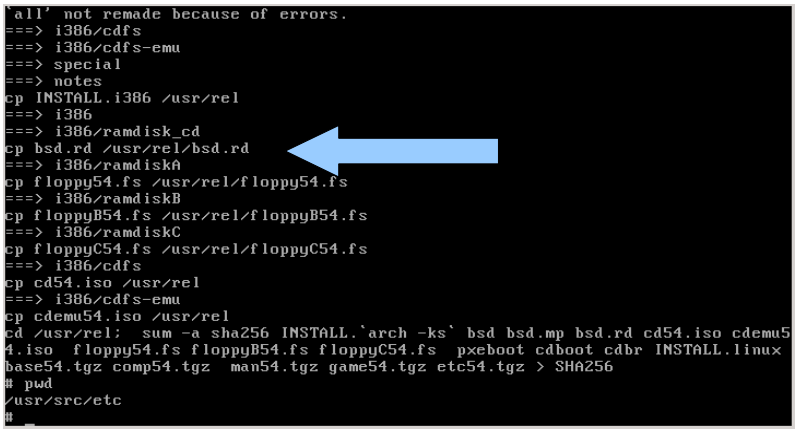

Illustration 7: Completed "make -k release" Build With Modified bsd.rd

The new bsd.rd (in /usr/rel or wherever you pointed your RELEASEDIR variable) now contains all the modifications needed.

Finally, move this file into your TFTP boot file location and you are ready to go.