Customizing ISOs and the Honeynet Project's Honeywall

David Dittrich *

March 22, 2004

Abstract

The concept of the honeywall goes back to discussions at CanSecWest CORE '01, where the idea was born to use a bootable CD-ROM to implement a layer 2 filtering bridge style device. The honeywall, as it was called, was designed with features to help make it easy for someone with little technical knowledge to set up a honeywall, but the standard distribution would not scale to a large site's needs. Using two types of customization capabilities, new programs can be added to the standard distribution ISO, and the way that the system configures itself at boot time can also be customized (including changing default passwords, adding SSH keys, etc.)

1 Introduction to the Honeywall

1.1 The History of the Honeywall

The Honeynet Project began development of a bootable CD-ROM that implements a layer-2 filtering bridge style firewall in 2003. This system — named the "honeywall" because it is a hybrid firewall/IDS system designed to implement honeynet data control and data capture — is based on a combination of a modified iptables rules, Snort[1] and snort-inline. These patches, scripts, and program source can be found in the tools section of the Honeynet Project web site[2].

The concept of the honeywall goes back to CanSecWest CORE '01[3], where several members of the Honeynet Project and others were discussing ways of making honeynets harder to detect and more flexible. Many of the Project members present were already using OpenBSD[4] as bridging firewalls[5] for personal use on networks at home and work, so this firewall implementation was easy to accept. Then the idea was born to implement this as a bootable CD-ROM, for a number of reasons.

One was to make it brain-dead easy for someone to set up a honeynet sensor. No more "which packages do I have to install?" No more manual configuration of applications you need to get a normal system up and running. The honeywall CD comes with pre-installed utilities, things that are needed enabled and things that aren't disabled, special tools already added, etc.

Another was to allow ultra rapid deployment of a honeynet sensor, for example to deal with an incident response scenario. As long as you can locate an x86 compatible PC with a ROM that can boot from a CD device, room for one or two NICS (if you are using a dual-port NIC and have one built-in NIC, you can get by with a single PCI or IDE slot), and a hard drive to hold collected data, you can simply put this device in-line between a suspected compromised host and its wall port and you have transparent inline logging capability.

1.2 The need for customization features

Around this same time, William Salusky created the *Biatchux* bootable CD-ROM forensic toolkit (now known as *FIRE*[6]). FIRE integrates a number of forensic tools in a handy bootable CD-ROM. It (and

*D. Dittrich: University of Washington, Seattle, WA.

ISBN 0-7803-9814-9/$10.00

all other such distributions) suffer at least one major drawback: It always booted to a clean and unconfigured state. Each time you started again, even if you did so on the same host without any other changes, you would have to go through the same steps of creating an IP address, perhaps generating new SSH keys, etc. There was no way to ensure the system booted up a certain way in a repeatable manner by setting things before burning it to CD-ROM.

Dittrich suggested a hack to Salusky that implements what Dittrich refers to as *holes* in the ISO image. These holes, which are overwritten in-situ before the ISO is burned to CD-R, allow for a custom boot sequence. (This concept will be described in more detail in a moment.)

Now the ISO had the flexibility to be customized; to come up and request a DHCP lease or set a static IP address, to have custom SSH public/private keys and authorization files, to have a custom password on the root account, even to swap the caps lock and control keys back they way the should be! Each time the CD was used to boot the system, it came up a predictable way and was usable remotely right from the start.

Of course some people won't need to enhance the ISO at all, which is why the decision was made to build in a simple user interface. That way the honeywall can be used in a stand-alone manner. Documentation on how to use it from the stock ISO will provide all that is necessary for someone to implement a stand-alone honeynet.

But there are many more interesting things that can be done if the sensor network is increased in size and dispersed widely, which requires a more careful, more flexible design, and richer feature set. To scale to a thousand honeywalls and configure them (or reconfigure them if you have to respond quickly to something like a newly publicized vulnerability in Windows RPC/DCOM, OpenSSL, or `rdist`) all within a short time window, requires that the systems need to not only be easy to set up and modular in design, but to also have facilities for customization for local network settings and centralized remote management.

2 The ISO 9660 CD-ROM file system

The Compact Disc has been around since 1980. It was first used widely as a new media for distributing music albums, but as prices of drives came down (and sizes of commercial operating systems and software programs increased), CDs also became popular for distribution of software. By April of 1998, the International Standards Organization (ISO) defined standard 9660, defining a CD file system format.[7] and a standard method for making a CD bootable.

Bootable CDs are created by embedding a floppy disk image in a fixed location, which the drive and system BIOS know how to extract and use as a boot device then allow the kernel to mount the remainder of the CD for the operating system to use.

Most people who are familiar with CD-R (CD recordable) or CD-RW (CD Read/Write) media today use them only to store large amounts (around 700MB total) of files or for installing software. The CD appears as just another removal drive to the operating system. Operating systems like Windows and the Macintosh OS hide the underlying processes of mastering and burning CDs, making their creation as easy as a drag and drop operation, so many people aren't aware of the underlying details. Even fewer would consider doing anything with ISO 9660 images except copying them around the network and burning them to CD.

2.1 Mastering an ISO 9660 image and burning a CD

Any set of files or directories can be turned into an ISO 9660 format file system image (known as an *ISO* for short[1]) Those using Unix can do this with the *mkisofs* [8] program. The result is a file that conforms to the ISO 9660 standard, and is ready to be stored in a web server for later retrieval. This is a common way for people to obtain and burn their own CD-ROMs

[1] Of course *ISO* stands for International Standards Organization, so its popular use to refer to a CD image is technically misleading. *ISO 9660 image* would be a more accurate term to use.

ISBN 0-7803-9814-9/$10.00

for installing operating systems, such as Linux.

Once copied to a local system, the ISO 9660 image can be written to CD media. This can be accomplished in Unix using the *cdrecord* [9] program [2]

There are two key facts that are not clear in the description so far (and aren't clear when reading the ISO 9660 specification either) that are central to customizing ISO 9660 images.

1. While the process of creating CDs as described above is a one-way, write-only process, ISO 9660 images can be taken apart almost as easily, turning them back into their constituent files. The files can then be modified and the process of mastering the ISO 9660 image can be repeated.

2. There is no integrity checking built into the ISO 9660 format. This means that as long as one is careful and respects the length boundaries of files within an ISO image, you can over-write any bytes you want within the files in an ISO 9660 image and change them before burning them to CD.

A variation of the first method is used popularly for customization of the KNOPPIX bootable Linux ISO. You can find documentation describing these steps at the KNOPPIX web site[12] and in a talk by Austin Godber[13].

While this process of re-mastering KNOPPIX ISOs works to create customized KNOPPIX ISO images, it is very labor intensive and would need to be done multiple times (modifying multiple files each time) to create multiple distinctly different ISO images. In other words, this solution does not scale. It can, however, be used as a way to create a basic template that can then be customized in a more flexible way, and that is the unique method described in this paper.

3 Honeywall Customization

There are two primary types of customization of the Honeywall ISO that a site would want to consider. These are called *template* (or *Type 1*) customization, and *boot-time configuration* (or *Type 2*) customization.

- **Template customization** can be viewed as modifying the files and directories that are stored in the ISO 9660 file system itself (which is not a read/write file system, so most people don't think of it as being something one can change.) In order to change the contents of the static ISO 9660 file system, it is necessary to rip that static file system apart, copy its contents to a staging area where they can be accessed and modified, then create a new ISO 9660 file system image from the staged files and any new files you want to add.

- **Boot-time configuration customization** uses the technique of selectively and carefully overwriting pre-placed files (*holes*) in the ISO — at least one of which is run as part of the boot sequence — with custom commands to control how the system boots up.

As their names imply, template customization (Type 1) affects the file system contents that form the operating system template that will be seen by the system when it is in a running state, and boot-time configuration customization (Type 2) affects boot options, variable settings, and other things during the boot process. [3]

A development host running Linux is required to customize the ISO. We'll call this system the *dev host*. In the directory where you will be customizing the ISO, staging directories are used for incoming files from the *original ISO*, and outgoing files to be made into a new *template ISO*. Both types of customization are accomplished using some scripts and `make` files that are bundled in the Honeywall Customization kit. We will assume that the reader has created a directory for customization and has changed the current working directory of their shell to this location (our example uses the directory `/usr/local/src/honeywall/customize`).

[2] All of these steps are documented in the Linux "CD writing tutorial."[10] Graphic front ends, such as CDRoast[11], hide these details and make it really easy to create data or media CDs in Linux.

[3] In reality the distinction between Type 1 and Type 2 customization is a little more complicated than is depicted here, but this explanation will suffice for now.

3.1 Template (Type 1) customization

The first kind of customization of the ISO is the addition, removal, or replacement of programs and files in the *original ISO* to produce the *template ISO*.

You would want to consider Type 1 customization if you wanted to add a set of new programs and hook into the User Interface to control them by modifying the existing UI scripts to include options to run your own programs (e.g., to add new data analysis features to the Status menu), or to modify one of the existing scripts to fix a bug or change a default option.

3.2 Example Type 1 customization

Let us look now at an example of how to augment the Honeywall ISO to include a new program, a UW modified version[14] of `tcpdstat`[15], and to add a new menu entry in *Status.sh* to support it.

Here are the steps you should follow:

1. Install the most recent release of the honeywall customization tools. In this example, we are using `honeywall-custom-52.tgz`.

```
# cd /usr/local/src/honeywall/customize
# wget http://staff.washington.edu/\
  dittrich/misc/honeywall-custom-52.tgz
# tar -xf /root/honeywall-custom-52.tgz
```

2. Copy the latest version of the honeywall ISO:

```
# cp ~/honeywall-0.65a.iso .
```

3. The customization tools use `make` and some scripts to configure and control everything. You first need to configure it to know about the name of the distributed Honeywall ISO you are going to use (in this case *honeywall-0.65a.iso*), the name you want to use for your template ISO (let's call it *hw-0.65a.iso*), the directories in which you will place files for Type 1 (*./root*) and Type 2 (*./root.local*) customization files, the name of the Honeywall configuration file you wish to use (default is *honeywall.conf*, and the IP address you will use for your Honeywall if you are doing live development testing.

   To configure, type `make configure` and answer the questions it asks:

```
# make configure
################################################################
You are about to (re)configure scripts used for enhancing
a stock Honeywall ISO and/or for customizing your own site-
specific honeywall ISO images.

Use Ctrl-C now if you do not wish to do this.
################################################################

Which ISO would you like to customize? [honeywall-0.65a.iso]:
What do you want to call the template ISO? [hw-0.65a.iso]:
Which directory holds programs to add to the ISO? [./root]:
Which directory holds files for customizing the ISO? [./root.local]:
Which import file would you like to use? [honeywall.conf]:
What IP address is your development test honeywall? [10.10.10.50]:

Configuring Hakefile
Configuring Hakefile.iso
Configuring isoutil
```

   In this example, we had already entered the name `honeywall-0.65a.iso` as the *original ISO*, `hw-0.65a.iso` for the name of our *template ISO*, etc. Once you have set these values, they are saved and will serve as defaults in future.

4. To get access to files on the Honeywall ISO that you wish to modify (in this case, */dlg/Status.sh*), use the command:

```
# make unpack
```

   You now have a file tree that contains all of the files (minus things in the initial ram disk) in the directory `./iso-in-stage/root`.

5. Copy all of the files you want to add to this honeywall ISO image into a sub-directory named `./root/` in the customization directory. This directory is something like that used in in a `chroot` environment, where the root of the run-time file system is just a subdirectory when viewed from "outside."

   The `find` command just shows the paths to files in `./root/` for illustration purposes.

```
# mkdir root
# (cd /usr/local; tar -cf - .) |
> (cd root; tar -xf -)
# find ./root -type f
 . . .
./root/usr/local/bin/tcpdstat
./root/usr/local/sbin/ntop
 . . .
```

6. Copy the script that implements the new feature you want, in this case to identify all Snort log files and provide a menu interface to select which one to obtain statistics on using *tcpdstat*[4] Before copying a file into a sub-directory within `./root`, it is wise to make sure it exists first.

```
# mkdir -p root/dlg
# wget http://staff.washington.edu/\
  dittrich/misc/tcpdstat.sh
# mv tcpdstat.sh root/dlg/
```

7. Now copy the *Status.sh* file from the staging area into the directory where your Type 1 customization files will be kept (that is, into `./root/dlg` in this case.)

```
# cp iso-in-stage/root/dlg/Status.sh \
  root/dlg/Status.sh
```

   You are now ready to edit the file.[5]

8. To add the files in `./root/` to the ISO, you use the following command:

```
# make template
```

   Make will inform you of what it is doing along the way. If it succeeds, you will see:

```
# make template
 . . .
Max brk space used a000
21040 extents written (41 Mb)
-rw-r--r--    1 root     root     43089920 Dec 13 18:51 hw-0.65a-beta.iso
30669e354f0d617faf6249b939599e8e  hw-0.65a-beta.iso
make[1]: Leaving directory `/w/honeywall/customize'
```

9. From this point on, as long as you aren't trying to add new programs to the original ISO, you can just use the Type 2 customization features to alter the boot configuration (which also allows a limited addition of files, provided they total less than 1MB.)

   This *template ISO* can now be customized for boot-time using the method described in the next section.

[4] A copy of a script to do this can be found at http://staff.washington.edu/dittrich/misc/tcpdstat.sh

[5] You can find a copy of *Status.sh* already modified to run the new */dlg/tcpdstat.sh* script, at http://staff.washington.edu/dittrich/misc/Status.sh

3.3 Boot-time (Type 2) customization

Recall that the Type 2 customization features are based on work done by David Dittrich and William Salusky for customizing the FIRE forensic CD. The hack that is used is a set of Bourne shell scripts that are nothing but a few thousand bytes (to over a megabyte) of comments. These are known as "*holes*" in the ISO. This same method of using holes is employed by the Honeywall ISO.

If you were to look at one of these holes — the file that makes up the hole, that is — in an uncustomized ISO, it would look something[6] like this:

```
#!/bin/bash
#<-- custom.sh -->
#-------------------------------------------;
#-------------------------------------------;
#-------------------------------------------;
#-------------------------------------------;
#-------------------------------------------;
#-------------------------------------------;
             [many lines removed]
#-------------------------------------------;
#-------------------------------------------;
```

The hole, in this case, is identified by the name "`custom.sh`" between the "`<--`" and "`-->`" strings.[7] A simple Perl script locates these "holes" in the ISO. It then checks to see if a file with a matching name exists in the customization directory, and fills them in with commands of the user's choosing (i.e., the contents of files with those same names in the customization directory.)

The boot-time control of how the system is configured is accomplished by having the *custom.sh* file be executed as part of the boot sequence. Since this is simply a script, and can

[6] These lines are shortened to fit on this page. In reality, the lines are 80 characters long.

[7] To be precise, these strings must also occur at the start of a line, and must immediately follow the pound-bang /bin/bash line. This prevents finding and filling a *description of a hole*, e.g., in a README file.

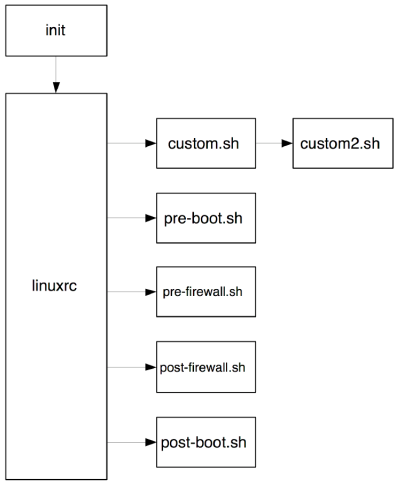

Figure 1: The execution path of scripts at boot.

be accompanied by inclusion of other scripts that you control, anything can be done at boot time. Anything that you can script, you can have the system do on boot. The sequence of execution of scripts in the boot process is depicted in Figure1. The arrows show execution paths from `init` → `linuxrc` → `custom.sh` and so forth for → `pre-boot.sh`, → `pre-firewall.sh`, → `post-firewall.sh`, and → `post-boot.sh`. (You can also see the relationship of `custom.sh` → `custom2.sh`.

This provides a way to customize things like switching caps lock and control keys, obtaining a DHCP lease (or defining a static IP address, netmask, host name/IP mappings, etc.), replacing the default root password, and loading in a limited set of new programs into writable areas in the system's RAM disk.

This technique provides a means of customizing the boot-time configuration of the CD, but has very limited ability to add new programs to the system. (Granted, you could keep adding more holes or make them bigger, but that simply uses up space in the ISO that may not be necessary.) This mechanism also suffers from the problem that you can't remove programs from a read-only compressed filesystem. You are only able to load files into RAM disk areas (which is why the previous method of customization is also required.)

3.4 Example Type 2 customization

Just like you placed files in a directory hierarchy named `./root/` to add new programs to the template ISO, you create a directory called `./root.local/` for adding custom configuration files such as those listed above. This way, your template ISO does not include these sensitive files, and can be given to someone else within your organization to customize on their own. (See section 4 for security considerations regarding customized honeywall ISOs and CD-ROMs.)

Some things you will want to consider for Type 2 customization (shown with the path relative to the customization directory, so you know where to put them) include:

- Custom passwords in `./root.local/etc/shadow` (Create this file by making a bogus account in `/etc/passwd` and changing its password, then editing the line to have "root" as the account name. It is advised to set your own default password to avoid someone using the default to access the honeywall before you change it manually.)
- Custom SSH keys in `./root.local/home/root/.ssh` (This simplifies remote administration by allowing remote command execution and tunneling using SSH.)
- Known SSH hosts keys and ssh client/server configuration files in `./root.local/etc/ssh`
- Customized swatch rules file in `./root.local/etc/swatchrc`

Here is what a typical directory hierarchy might look like:

```
# find root.local -type f
root.local/etc/ssh/sshd_config
root.local/etc/ssh/ssh_config
root.local/etc/ssh/ssh_host_rsa_key
root.local/etc/ssh/ssh_host_rsa_key.pub
root.local/etc/ssh/ssh_host_dsa_key
root.local/etc/ssh/ssh_host_dsa_key.pub
root.local/etc/shadow
root.local/etc/swatchrc
root.local/home/root/.ssh/id_dsa
root.local/home/root/.ssh/id_dsa.pub
root.local/home/root/.ssh/id_rsa
root.local/home/root/.ssh/id_rsa.pub
root.local/home/root/.ssh/fingerprints.txt
root.local/home/root/.ssh/identification
root.local/home/root/.ssh/authorized_keys
root.local/home/root/.ssh/known_hosts
```

Other customizations to control boot-time behavior can be done using one of the following "holes" (empty shell scripts, which are executed in this order at boot time: See Figure 1): `custom.sh`, `pre-boot.sh`, `pre-firewall.sh`, `post-boot.sh`, `post-firewall.sh`

3.5 Creating your customized CD-ROM or CD-RW

Once you have made all the additions, edits, customizations, etc., you are ready to burn your CD. Create a new CD-R from your template ISO and custom files in `./root.local/` using this command:

```
# make cdrom
```

(Substitute `cdrw` for `cdrom` if you wish to write a CD-RW instead of a CD-R. CD-RW is suggested if you are going to be doing this frequently for testing, as it saves on wasted plastic.)

You now have a customized honeywall CD ready to boot. To test the new feature that you just added to run *tcpdstat*, see the new final item in the Status menu.

4 Security Considerations

Lastly, we will examine some of the security considerations encountered when using a customized bootable CD.

4.1 Physical security

Using a bootable CD-ROM based operating system has some physical security issues that aren't *quite* as bad as with normal systems. At least you can protect workstations that normally boot from a hard drive by using BIOS passwords and pre-defined boot device selection to try to secure the hard drive (although someone could just walk off with the whole system and get access to it by opening the case.)

When booting from a CD-ROM as the primary boot device, it is relatively easy to get the drive door to open and eject the disc, which an attacker can then plop into another system and copy and/or analyze. Or someone may throw away an old ISO or have it "walk off" if left lying on a desk.

Best practice here is to lock the system into a physical enclosure so that nobody can get to it, physically destroy all old ISOs (or better yet, cycle CD-RW discs), and lock up any CDs that are not currently being used.

4.2 Passwords and SSH keys

The honeywall ISO is delivered with a default password. When you use the initialization interview in the user interface, it will ask you to change root's password. You can avoid needing to do this by using the customization features described in section 3.

It does not come with any SSH keys (you can generate them using the user interface if you want, but it may be simpler for large installations to have static keys that are automatically included in the ISO.)

4.3 File system security

The most vulnerable information are the secret password hashes, private SSH keys, etc., that you wish to have exist in the run-time file system. Since the CD-R/CD-RW is not encrypted, you must make *sure*

that nobody can get their hands on a copy. (This is another reason why there are two directories used in customization, `./root/` for new programs, and `./root.local/` for local files. That is, a site may distribute its scripts for adding new programs to the honeywall ISO, but may wish to keep all SSH keys and shadow files unique to each customized ISO.)

5 Conclusion

So in conclusion, we have seen the two types of configuration — *template customization* and *boot-time customization* — that are supported by the Honeywall CD (or more accurately by its distributed ISO 9660 image). Examples were provided that show the essential commands and features, including one concrete example of altering the Status menu. Further documentation that accompanies the Honeywall ISO and customization utilities provides more details, and a FAQ is available on the Honeynet Project's web site[16]. Readers who fix bugs or develop new features for the Honeywall using these techniques are encouraged to send them to the Honeynet Project as described in the Honeywall FAQ.

References

[1] M. Roesch, "The snort intrustion detection system." Web site http://www.snort.org/.

[2] T. H. Project, "Tools." Web site http://project.honeynet.org/tools/index.html.

[3] CanSecWest, "Core 01." Web site http://www.cansecwest.com/archives.html.

[4] OpenBSD, "The openbsd operating system." Web site http://www.openbsd.com/.

[5] D. Hogan and D. Bryan Hinton, "Openbsd bridge without ips using ipf tutorial." Web site http://www.daemonnews.org/200103/ipf_bridge.html.

[6] W. Salusky, "Fire bootable forensics cd." Web site http://fire.dmzs.com/.

[7] C. Summers, "Introduction to iso 9660." Web site http://www.singlix.com/trdos/IntroductionToISO9660.pdf.

[8] E. Youngdale, "mkisofs." Web site http://www.andante.org/mkisofs.html.

[9] J. Schilling, "cdrecord." Web site http://www.fokus.gmd.de/research/cc/glone/employees/joe

[10] W. Trmperm, "Linux cd writing tutorial." Web site http://wt.xpilot.org/publications/linux/howtos/cd-writing/html/, July 2000.

[11] T. Niederreiter, "xcdroast." Web site http://www.fh-muenchen.de/home/ze/rz/services/projects/xcdroast/e_0.96 readme.html, November 1998.

[12] K. web site, "Knoppix remastering howto." Web site http://www.knoppix.net/docs/index.php/KnoppixRemaster

[13] A. Godber, "Customizing knoppix." Web site http://uberhip.com/people/godber/plug, September 2003.

[14] D. Dittrich, "Uw modified tcpdstat." Web site http://staff.washington.edu/dittrich/tools/tcpdstat.tar.

[15] K. Cho, "tcpdstat." Web site http://www.csl.sony.co.jp/person/kjc/kjc/papers/freenix200

[16] T. H. Project, "Main web page." Web site http://project.honeynet.org/.

ISBN 0-7803-9814-9/$10.00