STANDARD OPERATING PROCEDURE

FOR ADDING A PACKAGE INTO LINUXWALL

CONTENTS

I. INTRODUCTION 2
II. FIND THE PACKAGE YOU NEED AND TRY IT 2
III. COMPILE THE PACKAGE WITH UCLIBC 2
IV. TRIM THE IMAGE SIZE 7
V. CONFIGURE THE PACKAGE 8
VI. COMMIT AND BUILD THE FINAL IMAGE 9
VII. TEST ITS OPERATION 11
VIII. FURTHER READING 12

DATE: 2004/06/14 VER: 0.8
DATE: 2004/10/06 VER: 0.9
DATE:2006/1/24 VER: 1.0
DATE:2006/10/4 VER: 1.1
DATE:2009/6/12 VER: 1.2

High Speed Network Lab.
Department of Computer Science
National ChiaoTung University, Hsinchu, Taiwan
http://speed.cis.nctu.edu.tw

I. INTRODUCTION

LinuxWall is a Linux distribution derived from the early version of BuildRoot. The derivative integrates many networking and security features/packages, so it is good to take the LinuxWall as the first step of building a security gateway[1]. This document gives the step-by-step procedures on how to compile and install the LinuxWall. It also teaches you the procedures of integrating a new package into LinuxWall. After reading the document and practicing all steps, you will learn how to construct an embedded system from scratch.

II. FIND THE PACKAGE YOU NEED AND TRY IT

a. Procedures

Step	Description	Notes
1	Write down the functionality that will be added into LinuxWall.	maybe one of "vpn", "anti spam", or "antivirus"
2	Search by Google with the keyword: name + "open source" and browse the official website.	- enter "vpn open source" in Google - URL of official websites may end with ".sourceforge.net" or ".org".
3	Select a suitable package and download it to the directory /usr/local/bin of your personal Linux system.	Full-installed **RedHat 9** is the recommended dev. system when you carry out all the procedures in this SOP.
4.	Unpack the package	tar xvfz xxxx.tar.gz
5	Configure and Install the package	./configure make make install
6.	Try its functions	

III. COMPILE THE PACKAGE WITH UCLIBC

a. Procedures

Step	Description	Notes
0	Install ReaHat 9 Install gcc-3.3 or gcc-3.4	Many porting efforts have to be done if you want to try LinuxWall in a newer OS or toolchain. ref.III.b.12

[1] Ying-Dar Lin, Huan-Yun Wei, Shao-Tang Yu, Building an Integrated Security Gateway: Mechanisms, Performance Evaluation, Implementation, and Research Issues, IEEE Communication Surveys and Tutorials, Vol.4, No.1, third quarter, 2002.

1	Understand what uClibc and toolchain are	http://www.uclibc.org/about.html http://www.uclibc.org/toolchains.html
2	Set the environment variable of svn.	ref. III.b.2
3	- Checkout the module LinuxWall from the SVN server. - (not recommended) You can also directly download the uClibc development environment from the uclibc website if you plan to get the pure uClibc development environment.	- `svn co $SITE/linuxwall` -Check the “Download” section in http://www.uclibc.org/ ref. III.b.11
4	Checkout the packages from svn and then build a soft link	1. `svn co $SITE/linuxwall_dl` 2. `cd linuxwall/sources` 3. `ln -s ../../linuxwall_dl dl`
5	make and then chroot to the LinuxWall image.	- Be root, because some packages, such as tripwire, need the root privilege. - Type “make” under the directory, linuxwall” - ref.III.b.12
6.	(Step 6~10 are required only when you try to integrate a new package into LinuxWall) Write make/xxx.mk to let buildroot know how to build your package.	you can ref make/gzip.mk or mke2fs.mk or openssh.mk
7.	Put necessary patchs in sources/	may need to patch the makefile or configuration originally offered by the package
8	Add one line into Makefile TARGETS += XXX	
9	Just run ‘make’ to build the image with the new package.	ref. III.b.7, III.b.8, III.b.9
10.	Chroot to the new image and test if the libs linked by the new package are correct.	You can use ‘ldd’ to check the lib linking states.

b. Guidelines

1. Current SVN server IP is 140.113.88.160
2. If your shell is TCSH

```
setenv SITE http://140.113.88.160/repos
```

if your shell is BASH

```
export SITE=http://140.113.88.160/repos
```

3. You can learn how to use subversion(SVN) from
   a. http://www.clear.rice.edu/comp314/svn.html
   b. http://aymanh.com/subversion-a-quick-tutorial

4. http://buildroot.uclibc.org/buildroot.html: a document introduce the tool for packaging the image and the structure in the xxxx.mk
5. The only way to learn how to write your xxx.mk is to study other *xxxx.mk* files. You can easily browse all xxx.mk files from http://www.uclibc.org/cgi-bin/cvsweb/buildroot/make/ The following information just offers you a limited assistance.

Table 1. Important variables referred in 'make/xxxxx.mk' but defined in 'Makefile'

BUILD_DIR	All downloaded packages are unzipped in this working dir.
TARGET_DIR	After compiling, the results should be copied to this directory.
SOURCE_DIR	All patch files are put in this dir.
DL_DIR	All downloaded package are put in this dir
STAGING_DIR	Libraries and header files in this dir can be used as building
TARGET_CROSS	The cross compiler
TARGET_CC	$(TARGET_CROSS)gcc
STRIP	The command to downsize the executing code

Table II. Important variables referred and defined in 'xxx.mk'

XXXXX_SITE	Tell buildroot where to download the package
XXXXX _DIR	Tell buildroot where to be the working dir
XXXXX _SOURCE	the tar file name of the package
XXXXX _PATCH	the path and filename of patch file (optional)

6. Five main parts are usually included in xxx.mk. make/openssh.mk is a good sample.
   - unpacked
   - configured
   - strip: downsize the code
   - make and install
   - how to clean the built results or intermediate files.
7. Remember to add
   $(TARGET_CONFIGURE_OPTS) \
   LD=$(TARGET_CROSS)gcc \
   CFLAGS="$(TARGET_CFLAGS)"
   in the 'configured' part and set
   CC=$(TARGET_CC),
   as you mean to make the package.
8. If you CANNOT compile the package for missing some necessary libs, build the missing libs as you build a package, and then copy them into the dir specified by $(STAGING_DIR).
9. If you CANNOT make the linuxwall and see the following msg
   error: cannot check ..... when cross compiling
   , then you should `vi config.log` to know which line in the file `configure` leading to the failure. The following are possible codes leading to the failure.
   ```
   if test "$cross_compiling" = yes; then
     { { echo "$as_me:14201: error: cannot run test program while cross compiling" >&5
   ```
   The solution is to remove these checks. For example,
   ```
   if test 0 = 1; then
     { { echo "$as_me:14201: error: cannot run test program while cross compiling" >&5
   ```
10. If you CANNOT cross-compile the package finally, a possible solution is offered as follows. First, you can chroot to the prebuilt buildroot development image and build the package in this environment. Then, pack the results to a tar file. Finally, in xxx.mk you just ask buildroot to unpack the tar file into the target dirs. Such a solution is adopted as we build Perl. You can refer perl.mk for further information.
11. If you want to use the latest uClibc development environment, the following steps are recommended: 1) Download and buildroot package and uncompress it, says, to directory DIR_B; 2) compile the buildroot package; 3) Modify the values of KERNEL_CROSS and TARGET_CROSS in the Makefile under linuxwall/. The new values should be like "DIR_B/ build_i686/staging_dir/usr/bin/i686-linux-uclibc-" if your target architecture is

i686.

12. Some problems you might meet:

a) Error about current_menu raises when compiling build_i386/linux-2.6.6/scripts/kconfig/mconf.c. The solution is to comment out the declaration of current_menu in that file.

b) Error about md_relax_table raises when compiling toolchain_build_i386/binutils-2.14.90.0.7/gas/config/tc-i386.h. The solution is to comment out its declaration in that file, and add its declaration in toolchain_build_i386/binutils-2.14.90.0.7/gas/tc.h.

c) Errors about no directory "ld" raises when making toolchain_build_i386/binutils-build. The solution is to comment out the "cd ld" line in the "install-ld" subsection in toolchain_build_i386/binutils-build/Makefile. Note that you also have to trim the backslash of its previous line, i.e., "$(SET_LIB_PATH)."

d) Errors about lvalue raises when making toolchain_build_i386/gcc-3.3.3/gcc/read-rtl.c. The solution is to modify the macro, obstack_ptr_grow, in toolchain_build_i386/gcc-3.3.3/include/obstack.h.

Before modification:

```
*((void **)__o->next_free)++ = ((void *)datum);                    \
```

After modification:

```
*((void **)__o->next_free) = ((void *)datum);                      \
 (__o->next_free)+=sizeof(void *);                                 \
```

e) Errors about no file, limits.h, raises when compiling __assert.c. The solution is to copy limits.h from the directory, include-fixed, in your uClibc, e.g., DIR_B/build_i686/staging_dir/usr/lib/gcc/i686-linux-uclibc/4.3.3/include-fixed.

f) Error happens while making "file." The version of /usr/bin/file utility in your system might be too newer to compile the magic utility. To solve it, you can modify magic.mgc subsection in the Makefile under build_i386/file-4.08/magic, and replace /usr/bin/file with $dir_linuxwall/build_i386/file-4.08/src/file.

g) Error of undefined reference to 'fatal' when compiling tripwire: Replace bs_htonl and bs_ntohl with htonl and ntohl in build_i386/tripwire-1.2/src/utils.c and build_i386/tripwire-1.2/sigs/snefru/snefru.c respectively.

h) Error happens when compiling squid. There is an unnecessary character "i" just one line before the reported error line, so remove the "i."

IV. TRIM THE IMAGE SIZE

a. Procedures

Step	Description	Notes
1	Just copy the compiled results to the specific directory, but not run ‘make install’ provided by the package.	Such a method can avoid the copying of help documents
2	Use STRIP to downsize executing file	
3	linking with shared libraries is suggested if the linking approach is provided by OS.	

b. Guideline

1. Reduce the spare space in a built image of file system。

   To reduce the spare space existing in a built image of file system, you can modify the parameter GENEXT2_ADDTOROOTSIZE in the file “ext2root.mk”. For example, you can change the parameter from its default 16384 bytes to 4096 bytes. Then, the built image will be reduced by 12KB.

   Notably, we have changed the block size used in the file system in order to reduce the image. However, according to our testing result, adjusting the block size brings a minor effect on the image size.

2. Remove the documents

   When you install a package, some documents like readme and user manual are usually copied into the image. However, these documents are unnecessary for system operation. Thus, you can remove these documents to reduce the image. To remove them, you should modify the makefile of the package by adding some commands. For example, in the makefile of the package squid linuxwall/make/squid.mk, a segment of commands are given after the configuration procedure, shown as

```
$(TARGET_DIR)/usr/local/squid/sbin/squid: $(SQUID_DIR)/src/squid
        -$(STRIP) --strip-unneeded $(SQUID_DIR)/src/squidclient
        -$(STRIP) --strip-unneeded $(SQUID_DIR)/src/squid
        -$(STRIP) --strip-unneeded $(SQUID_DIR)/scripts/RunAccel
        -$(STRIP) --strip-unneeded $(SQUID_DIR)/scripts/RunCache
        $(MAKE) -i -C $(SQUID_DIR) -l$(LIB_PATH)/crypt install;
        rm –rf    $(TARGET_DIR)/man
    cp $(SQUID_DIR)/src/squid.conf.default $(TARGET_DIR)/etc/squid.conf
    cp $(SQUID_DIR)/src/mime.conf.default $(TARGET_DIR)/etc/mime.conf
```

   The command -$(STRIP) --strip-unneeded is used to remove the files generated during the building procedure, such as .note or .comment. Besides, a

directory man and some help files are created during the procedure. Because we don't need these files to operate the system, we should delete the directory by the command rm to avoid the directory and files from being copied into the image.

Similarly, when you install other packages, you should check whether the unused files such as doc, man, example and info, are copied into the image. If these file are copied actually, you can remove them by the command mentioned above.

3. Remove the header file and the static-link library.

When you install a package, you need to avoid the header files and the static-link libraries from being copied into the image, because these files are unused in system operation. Generally, the header files are given in the directory "include". To remove them, you can use the command "rm -rf $(TARGET_DIR)/include/*.h". Similarly, you can use the command "rm -rf $(TARGET_DIR)/lib/*.a" to remove all the static-link libraries.

V. CONFIGURE THE PACKAGE

a. Procedures

Step	Description	Notes
1	(Step 1~2 are required only when you try to integrate a new package into LinuxWall) The configuration files of the package should be put in /sources/customize/etc	
2	Configure the package to let it run at booting if necessary.	/etc/rc.d/init.d
3	Provide a web GUI to start the package. you can put the php in /sources/customize/www.	
4.	You can put all files you want to copy into the final image in /sources/customize	Buildroot will automatically mirror all files and dirs in source/customer/ to the dir: root before packing all files to an image.

VI. COMMIT AND BUILD FINAL IMAGE

a. Procedures

Step	Description	Notes
1	Build a kernel that can be booted without initrd	ref. VI.b.1
2.	fdisk a clean disk to 2 partitions	If you are using VMWare, please reference VI.b.2 Command: fdisk ref. VI.b.3 One for kernel and one for rootfs.i386, e.g., /dev/sdb1 and /dev/sdb2
3.	format and mount the 2 partitions	mke2fs /dev/sdb1 mke2fs /dev/sdb2 mkdir /mnt/wall_kernel mkdir /mnt/wall_image mount /dev/sdb1 /mnt/wall_kernel mount /dev/sdb2 /mnt/wall_image
4	mount imgfile rootfs.i386	mkdir /mnt/tmp mount –o loop root_fs_i386 /mnt/tmp
5	Copy kernel to the boot disk	mkdir /mnt/wall_kernel/boot cp bzImage /mnt/wall_kernel/boot ref. VI.b.4
6	Copy root file system	cp –af /mnt/tmp/* /mnt/wall_image
7	install boot loader	mkdir /mnt/wall_kernel/boot/grub cp /boot/grub/* /mnt/wall_kernel/boot/grub cd /mnt/wall_kernel/boot/grub vi menu.lst ref. VI.b.5 rm ./grub.conf

		ln –s menu.lst grub.conf grub-install --root-directory=/mnt/wall_kernel /dev/sdb1 ref. VI.b.6
8	Umount all modified partitions	cd / umount /mnt/wall_kernel umount /mnt/wall_image

b. Guidelines

1. The kernel source code is under $your_linuxwall/buid_i386/linux-2.6.6/, when the target platform is i386.

```
cd $your_linuxwall/buid_i386/linux-2.6.6
make clean
make config       # you can also try “make menuconfig”
                  # all answers except the following ones should NOT be changed
    Enable loadable module support -> n
    SCSI device support -> y        # if the LinuxWall will be run in a SCSI hard disk, e.g., under VMWare environment
    SCSI disk support -> y
    BusLogic SCSI support -> y
make
```

2.(If you are using VMWare)

a. Suppose the virtual machine of your RedHat 9 is named as “RedHat 9.”
b. Edit the settings of “RedHat 9”
c. Add a new SCSI Hard Disk
d. Power on “RedHat 9,” and continue the remaining steps in V.a
e. After finishing the last steps in V.a, suspend “RedHat 9.”
f. Create a new virtual machine named as “Linuxwall”
g. Edit the settings of “Linuxwall”
h. Remove the default SCSI Hard Disk
i. Add an existing SCSI Hard Dist by browsing the directory and choosing the one you created in step c.
j. Power on “Linuxwall”

3. Google “fdisk linux” for more help.
   If you are too lazy to google it, then just type:
      fdisk /dev/sdb
         n, ENTER, p, ENTER, 1, ENTER,
         n, ENTER, p, ENTER, 2, ENTER,
         a, ENTER, 1, ENTER
         w, ENTER

4. bzImage is under $your_linuxwall/buid_i386/linux-2.6.6/arch/i386/boot/bzImage, when the target platform is i386.

5. You can modify menu.lst based on the existing menu.lst
   After modifying, it may look like,

(~skip~)
   title Linuxwall
      root (hd,0)
      kernel /boot/bzImage ro root=/dev/sda2

6. You may need to execute,

   grub-install --recheck --root-directory=/mnt/wall_kernel /dev/sdb

VII. TEST ITS OPERATION

a. Procedures

Step	Description	Notes
1	(Step 1~2 are required only when you try to integrate a new package into LinuxWall) Show the package can be executed at booting or by run a single script.	
2	Give a significant demo case for this package.	
3	(Step 3~4 are **not recommended**, if your demo case is a practice of LnuxWall SOP) Write down the demo case in the file /buildroot/democases/xxxx.demo	
4	Commit your change to svn server	

VIII. Further Reading

Buzybox: http://busybox.net
Buildroot: http://buildroot.uclibc.org/
uClinux: http://www.uclinux.org/index.html
SVN: http://durak.org/sean/pubs/software/version-control-with-subversion-1.6/