

## *Opening the Gilded Cage*

*Installing Apache + IMAP4 + PHP +  
Squirrelmail on a production server*

**Prof Joe R. Doupnik**

**Utah State University, and**

**Oxford University Computing Services**

**jrd@cc.usu.edu**

**joe.doupnik@oucs.ox.ac.uk**



## *Prologue*

**There comes a time in each system manager's quarterly report when he/she wishes to offer more services than can be done now, at no extra cost**

**There comes a time when an important software tool has evolved to have attributes deemed important, but the vendor's RPM is still four revs behind ("stable") or lacks that configuration option**

**What are we going to do about it?**

# *The goals*

**We wish to create our own versions of selected major applications**

**Use our own choices for configuration and features**

**Stay up to date at our own pace**

**Keep them free of the Linux vendor's material**

**Do not interfere with the Linux vendor's material**

## *Benefits and costs*

- + Our server behaves as we wish, many choices**
- + Least crosstalk with Linux vendor modifications**
- + Change rate is as we wish, can be within hours of a fixed release of a supporting/main product**
- Must make choices, do initial digging to resolve dependencies, correct installation problems**
- Need to monitor lists for bug reports**
- Need to keep clear notes on how an application is built our way**
- ± Vendor's configuration scripts will not apply, they can't cover all available application choices and they point at the vendor's installation files**

## *Things to keep in mind*

**Linux distributions from major vendors are done as “binary-only” - applications are precompiled by the vendor, their way, and offered as binary RPMs**

**What we see is what we get**

**Source RPMs (SRPM) are available too, with the material needed to reproduce the above RPMs**

**Building materials are separate from normal RPMs. Development RPMs have files needed to build other material from sources**

**We are about to build things ourselves, our way**

## *The tactical goals*

**Run popular webmail program Squirrelmail over Apache v2.latest**

**Requires Apache 2, IMAP4, PHP 4, Squirrelmail**

**Use SSL and create a certificate for Apache**

**Will require ferreting of other support components**

**Will require configuring inetd.d and xinetd.d scripts**

**We consider a major app to be Apache, PHP, and Squirrelmail.**

**Use binary RPMs for IMAP4 and OpenSSL . IMAP4 requires vendor additions to build our needed libraries.**

## *The tactical goals*

**Install our applications into /usr/local**

**The operating system does not store things there**

**This keeps our programs separate from similar editions maintained by the Linux vendor**

**Original sources may expect this placement of files**

**Turn off vendor apps in /etc/init.d scripts, and add our own application startup scripts there**

**Do not let our work appear in list of installed RPMs, else up2date/YaST may replace ours with vendor's**

# *Configuration aids*

**When building a major application we often must run app-specific script *configure* with many options**

**To simplify and record this, create a simple shell script invoking *configure* with those options**

**Create a text file of notes on other steps needed**



## *Getting ready*

**Unpack the sources to a safe spot (/home/app here)**

**Review the README and INSTALL files**

**Review the Makefile for special situations**

**Review *./configure* --help for available options**

**Consider final storage directory, say /usr/local, and specify that as the --prefix=/usr/local option to *configure***

**Usual installation steps are**

***./configure* --options, make, make install**

**We run *configure* via a shell script, to record options**

# *Apache*

**Fetch the current Apache v2 “tar ball” from  
<http://www.apache.org/>**

**Store it in say /home/apache**

**Unzip it, untar it there**

**cd httpd-2.0.xx**

**Explore: ./configure --help to see build options**

**For the lab, use shell script *myway.prefork***

## Looking at `./configure --help`

### Installation directories:

```
--prefix=PREFIX      install architecture-independent files in PREFIX  
                      [/usr/local/apache2]  
--exec-prefix=EPREFIX install architecture-dependent files in EPREFIX  
                      [PREFIX]
```

By default, 'make install' will install all the files in  
'/usr/local/apache2/bin', '/usr/local/apache2/lib' etc. You can specify  
an installation prefix other than '/usr/local/apache2' using '--prefix',  
for instance '--prefix=\$HOME'.

For better control, use the options below.

### ... many options not shown

### Fine tuning of the installation directories:

```
--bindir=DIR          user executables [EPREFIX/bin]  
--sbindir=DIR         system admin executables [EPREFIX/sbin]  
--libexecdir=DIR      program executables [EPREFIX/libexec]  
--datadir=DIR         read-only architecture-independent data [PREFIX/share]  
--sysconfdir=DIR      read-only single-machine data [PREFIX/etc]  
--sharedstatedir=DIR  modifiable architecture-independent data [PREFIX/com]  
--localstatedir=DIR   modifiable single-machine data [PREFIX/var]  
--libdir=DIR          object code libraries [EPREFIX/lib]  
--includedir=DIR      C header files [PREFIX/include]  
--oldincludedir=DIR   C header files for non-gcc [/usr/include]  
--infodir=DIR         info documentation [PREFIX/info]  
--mandir=DIR          man documentation [PREFIX/man]
```

## *Looking at ./configure --help*

Some influential environment variables:

CC	C compiler command
CFLAGS	C compiler flags
LDFLAGS	linker flags, e.g. -L<lib dir> if you have libraries in a nonstandard directory <lib dir>
CPPFLAGS	C/C++ preprocessor flags, e.g. -I<include dir> if you have headers in a nonstandard directory <include dir>
CPP	C preprocessor

Use these variables to override the choices made by 'configure' or to help it to find libraries and programs with nonstandard names/locations.

**We may need CFLAGS and LDFLAGS**

**Each application has its own *configure* file, if any**

## *Apache script myway.prefork*

```
$ cat myway.prefork
#!/bin/sh
./configure \
--prefix=/usr/local/apache2 \
--exec-prefix=/usr/local/apache2 \
--with-MPM=prefork \
--enable-mods-shared=all \
--enable-auth-digest \
--enable-mime-magic \
--enable-cern-meta \
--enable-charset-lite \
--enable-expires \
--enable-headers \
--enable-unique-id \
--enable-vhost-alias \
--enable-ssl \
--enable-dav \
--enable-info \
--enable-cgi \
--enable-dav-fs \
--enable-speling \
--enable-rewrite
```

This script is part of  
our documentation

There are many more choices possible. These will do for now.

## *Starting configuration*

**chmod a+x myway.prefork      to make it executable**  
**./myway.prefork                      execute it**

**There may be failures, we need to deal with each**  
**Look at error messages; choose troublesome item**

**If a header (.h) file is not found then it is in another**  
**location, or it is not present. Find out which.**

**find / -name foobar.h      look over entire file system**

## *Missing files*

**If the file is not present then see if an RPM has it  
Visit the Linux vendor's site, explore its RPM list,  
especially app-devel-xxx.RPM kinds.**

**Choose RPM or build unit from original sources**

**If a .h file is located in an unexpected spot then  
Add an include path for the C compiler, by typing this  
`CFLAGS=-I/this/is/the/place`  
`export CFLAGS`**

**Or edit *configure* to use the proper directory (ugh!)**

# *Unexpected file locations*

***Configure*** may expect a .h file and a library file to be in the same directory hierarchy, but our Linux vendor separates them

A solution is to edit *configure* to use our existing directories; another is rebuild the helper app

Some Linux vendors have embellished common worker apps such that we must add further items to our *configure* script myway.prefork. Kerberos is added by RedHat, as an example.

Patching *configure* is not uncommon, don't worry, but we try to avoid it



## *When we really get stuck*

**In some circumstances we do not have a worker application and we cannot find its source**

**Try <http://www.google.com/>, Linux list servers, etc**

**If the app is simply not available, or building it from scratch becomes unreasonable, then we must omit it from our list of options, perhaps until later research produces results**

**find and grep -r are our friends**

## *Apache, continued*

**When myway.prefork completes without error then  
build Apache via `make` and `make install`**

**After `make` please check for rationality:**

**`ls -l modules/ssl`**

**Expect to see many `.o` files as evidence of building**

**If this is successful then**

**`make install`**

**Copies built material to `/usr/local/apache2` etc**

**(does not overwrite pre-existing configuration files)**

# *Apache, SSL Certificate*

**Here there be tigers**

**SSL configuration is an opaque puzzle**

**There are three approaches**

- 1. Cheat and create with minimal info**
- 2. Slog through creating Certificate Authority et al.**
- 3. Understand this stuff and do it properly**

**We show the first two**

**Before starting, make these two subdirectories:**

**mkdir /usr/local/apache2/conf/~~ssl.crt~~**

**mkdir /usr/local/apache2/conf/~~ssl.key~~**

## *Making and using a Cert Auth*

**cd /usr/share/ssl/misc**

**cp CA myCA (or CA.sh to myCA.sh)**

CA is a shell script

**vi myCA**                      **change expiration days to longer, say 3365**

**./myCA -newca**              **create new Cert Authority**

**Answer questions, cn must be IP name or number**

**./myCA -newreq**            **cert request, answer same way**

**./myCA -signreq**          **CA signs our cert request**

**File newcert.pem is server's certificate**

**File newreq.pem is server's keys**

Here we are already tailoring a systems parameter: certificate life time

## *Making and using a Cert Auth*

```
cp newcert.pem /usr/local/apache2/conf/ssl.crt/server.crt
```

```
cp newreq.pem /usr/local/apache2/conf/ssl.key/server.key.org
```

```
cd /usr/local/apache2/conf/ssl.key
```

```
openssl rsa -in server.key.org -out server.key
```

(removes protective login when obtaining key at Apache startup)

```
chmod 400 server.key
```

protect plaintext key file

# ***Making and using a Cert Auth***

**Certificate Authority is /usr/share/ssl/misc/demoCA**

**The CA's private key is within demoCA/private**

**We can delete demoCA and contents if we wish to  
make a new CA**

**We can delete newreq.pem and newcert.pem**

Reference: <http://www.openssl.org/>

# *Apache2 quicky, two long lines*

```
openssl genrsa -rand /  
  proc/apm:/proc/cpuinfo:/proc/dma:/proc/filesys  
  tems:/proc/interrupts:/proc/ioports:/proc/pci:/proc  
  /rtc:/proc/uptime 1024 > /  
  usr/local/apache2/conf/ssl.key/server.key
```

Reads “stuff” from /proc to get lots of numbers for random generator

```
openssl req -new -key /  
  usr/local/apache2/conf/ssl.key/ server.key -x509  
  -days 3365 -out /  
  usr/local/apache2/conf/ssl.crt/server.crt
```

## *Apache2, ssl.conf touchup*

**Edit ssl.conf in the Apache conf directory**

**Ensure ServerName is the IP name of the machine**

**Touch up other items in that block:**

```
<VirtualHost _default_:443>
```

```
#    General setup for the virtual host
```

```
→ DocumentRoot "/usr/local/apache2/htdocs"
```

```
→ ServerName netlab3.usu.edu:443
```

```
ServerAdmin jrd@netlab2.usu.edu
```

```
ErrorLog /usr/local/apache2/logs/error.log
```

```
TransferLog /usr/local/apache2/logs/access.log
```



## IMAP4

**Squirrelmail and PHP will need IMAP4 email support**

**The popular material is from the Univ of Washington**

**Check installed RPM list for imap**

**rpm -q -a | grep imap** (query all rpms)

```
# rpm -q -a | grep imap
imap-devel-2002d-53
imap-lib-2002d-53
imap-2002d-53
```

**If we do not have it yet, then install this worker as-is. We do need the -devel- material.**

# *Configuring IMAP4*

**IMAP4 and POP3 components run upon demand via super daemon xinetd**

**Edit IMAP and POP scripts found in `/etc/xinetd.d` to enable these services**

**Restart xinetd to have them become functional  
`/etc/init.d/xinetd restart`**

**Don't forget your IP filters on these ports**

## *PHP version 4*

**Create directory /home/php**

**cd /home/php**

**mkdir /usr/local/etc            to hold PHP startup info**

**Download the current PHP v4 tarball from**

**<http://www.php.net/>**

**Unzip it, untar it**

**cd php-4.3.4                    or whatever version**

**Use shell script *myway* to configure it**

## *PHP v4 shell script myway*

```
# pwd
/home/php/php-4.3.4
# cat myway
#!/bin/sh
./configure \
--with-apxs2=/usr/local/apache2/bin/apxs \
--with-config-file-path=/usr/local/etc \
--with-openssl \
--with-zlib \
--enable-calendar \
--with-dom=/usr/lib/libxml2.so \
--with-zlib-dir=/usr/lib \
--enable-ftp \
--with-iconv \
--with-imap \
--with-imap-ssl \
--with-ldap \
--with-mcal \
--with-mcrypt \
--with-gettext \
--enable-sockets
```

**A really large number of choices exist. These will suffice for us.**

## *PHP building*

**We build PHP to be run as a cooperative interpreter within Apache, not as a cgi-bin style script**

**This method requires we build a PHP module loadable by Apache, and that requires the Apache module compiler `apxs2`**

**Results are stored in `/usr/local/apache2/modules`, and in `/usr/local/etc`**

**This is why we build and install Apache first  
`./myway` to get started, expect blockages**

## *PHP building blockages*

### First blockage:

```
checking whether to enable DBA interface... no
checking whether to enable dbase support... no
checking whether to enable dbx support... no
checking whether to enable direct I/O support... no
checking for DOM support... yes
not found
→ configure: error: Please reinstall the libxml2 >= 2.4.14 distribution
#
```

**rpm -q -a | grep xml** shows we have libxml2-2.5.10-32.RPM

**Discover we do not have, but need, libxml2-devel-2.5.10-25.RPM**

**Install it from Linux vendor (can update it later)**

**We guessed the devel RPM supplies a missing piece**  
**Re-run ./configure, blockage is solved**

## *PHP, next blockage ldap.h*

```
checking for IRCG support... no
checking for Java support... no
checking for LDAP support... yes
→ configure: error: Cannot find ldap.h
#
Thus we need openldap2-devel.rpm, not on system.
Fetch and install it.
Installation wants to add this too:
a+  cyrus-sasl-devel2.1.15                Cyrus SASL API
That worked.
```

**LDAP is installed, but the `-devel-` RPM is needed for `ldap.h`**

Play same `-devel-` hunch again

## *PHP next blockage mcal*

```
checking for external libmbfl... no
checking for MCAL support... yes
→ configure: error: Unable to locate your libmcal header files - mcal.h should be
in the directory you specify or in the include/ subdirectory below it - default
search location is /usr/local
#

Checking, no -devel lib. Looking for mcal.h shows it in /usr/include/mcal.
./configure says add directory, like this:
--with-mcal=/usr/include/mcal \
Now the blockage says:
checking for external libmbfl... no
checking for MCAL support... yes
→ configure: error: Unable to locate your libmcal library files - libmcal.a should
be in the directory you specify or in the lib/ subdirectory below it - default
search location is /usr/local
#
```

**Fix first error by saying --with-mcal=/usr/include/mcal in myway**

**./configure --help, read everything for hints**



## PHP, mcal continued

```
#  
However, libmcal.a is in /usr/lib. Thus there is a splitting of components  
involved. We need to say --with-mcal=/usr/include/mcal to find mcal.h. But  
configure wants libmcal.a to be underneath that directory, and it is not.  
To fix this edit configure to add /usr/lib to its search path for libmcal.a.  
That worked perfectly.
```

This one took some digging. Add **/usr/lib** to the search path for libmcal.a

```
→ Add /usr/lib  
for i in $MCAL_DIR $MCAL_DIR/mcal $MCAL_DIR/mcal/lib $MCAL_DIR/lib/mcal $MCAL_  
DIR/lib /usr/lib; do  
    if test -r "$i/libmcal.a"; then  
        MCAL_LIBRARY=$i  
    fi  
done  
  
for i in mcal cal_misc icalroutines; do  
"configure" 94116 lines --49%--  
46852,0-1 49%
```

## *The mcal case as canonical*

**mcal is a small third party application, designed to install into /usr/local/include/mcal & /usr/local/lib**

**Regular Unix systems place third party apps outside the main body of o/s material. Linux mixes everything together, splitting heavily.**

**mcal components were split into two systems areas and *configure* could not deal with the split**

**Splitting is the real problem**

**We could have built mcal from sources and eliminated the need to modify PHP's *configure***

**Duplicating mcal into vendor and unsplit /usr/local editions would be wise over the long term**

## *Don't raise the bridge...*

To avoid editing *configure* we can build libmcal from original sources:

<http://www.sourceforge.net/projects/libmcal>

If we do that and change *myway* to use just

`--with-mcal` with no qualifier then there is no error from mcal material (things are where *configure* expects them)

Thus duplicating the mcal installation solves a messy problem at almost no cost

```
# ls /usr/local
.  apache2  etc      include  man      sbin      squirrelmail
.. bin       games    lib       mrtg     share     src
# ls /usr/local/lib
.  ..  libmcal.a  libmcal.so  php
# ls /usr/local/include
.  ..  mcal  php
```

## *PHP, last blockage mcrypt.h*

```
checking for MCAL support... yes
checking for mcrypt support... yes
→ configure: error: mcrypt.h not found. Please reinstall libmcrypt.
#
We have libmcrypt but not -devel, thusly:
# rpm -q -a |grep crypt
libxcrypt-2.0-32
cryptplug-0.3.16-87
libmcrypt-2.5.7-33
libgcrypt-1.1.12-91

Install libmcrypt-devel from archives.
That also worked.
./myway completed without error.
```

We have this trick down cold now

# ***PHP building, completion***

**After myway finishes without error do**

**make**

**make install**

**cp php.ini-recommended /usr/local/etc/php.ini**

**Edit /usr/local/apache2/conf/httpd.conf to enable  
PHP sensitivity, shown on next slides**

Reference: <http://www.php.net/>

# *Apache httpd.conf for PHP*

**libphp4.so should appear automatically from PHP install**

```
LoadModule dir_module modules/mod_dir.so
LoadModule imap_module modules/mod_imap.so
LoadModule actions_module modules/mod_actions.so
LoadModule speling_module modules/mod_speling.so
LoadModule userdir_module modules/mod_userdir.so
LoadModule alias_module modules/mod_alias.so
LoadModule rewrite_module modules/mod_rewrite.so
→ LoadModule php4_module modules/libphp4.so
```

Hint: start Apache with SSL by

```
/usr/local/apache2/bin/apachectl startssl
```



## *Sense index.php as a dir-index*

```
#  
# DirectoryIndex: sets the file that Apache will serve if a directory  
# is requested.  
#  
# The index.html.var file (a type-map) is used to deliver content-  
# negotiated documents. The Multiviews Option can be used for the  
# same purpose, but it is much slower.  
#  
DirectoryIndex index.html index.html.var index.php
```

```
# AddType allows you to add to or override the MIME configuration  
# file mime.types for specific file types.  
#  
#AddType application/x-tar .tgz  
AddType application/x-httpd-php .php  
AddType application/x-httpd-php-source .phps  
#
```

Add these two lines manually so the PHP interpreter runs

## *Squirrelmail*

**Obtain Squirrelmail/latest tarball from  
<http://www.squirrelmail.org/>**

**Unpack it in /usr/local, creates its own directory**

**Rename directory to be squirrelmail**

**cd squirrelmail**

**chown -R myself \***                      **myself is a local user**

**chown nobody data**                      **user nobody is webserver**

**cd config**

**./conf.pl**                      **make local changes**

**chown nobody config.php**                      **for web based admin**

**No ./configure, no make, no make install**



## *Squirrelmail, Apache httpd.conf*

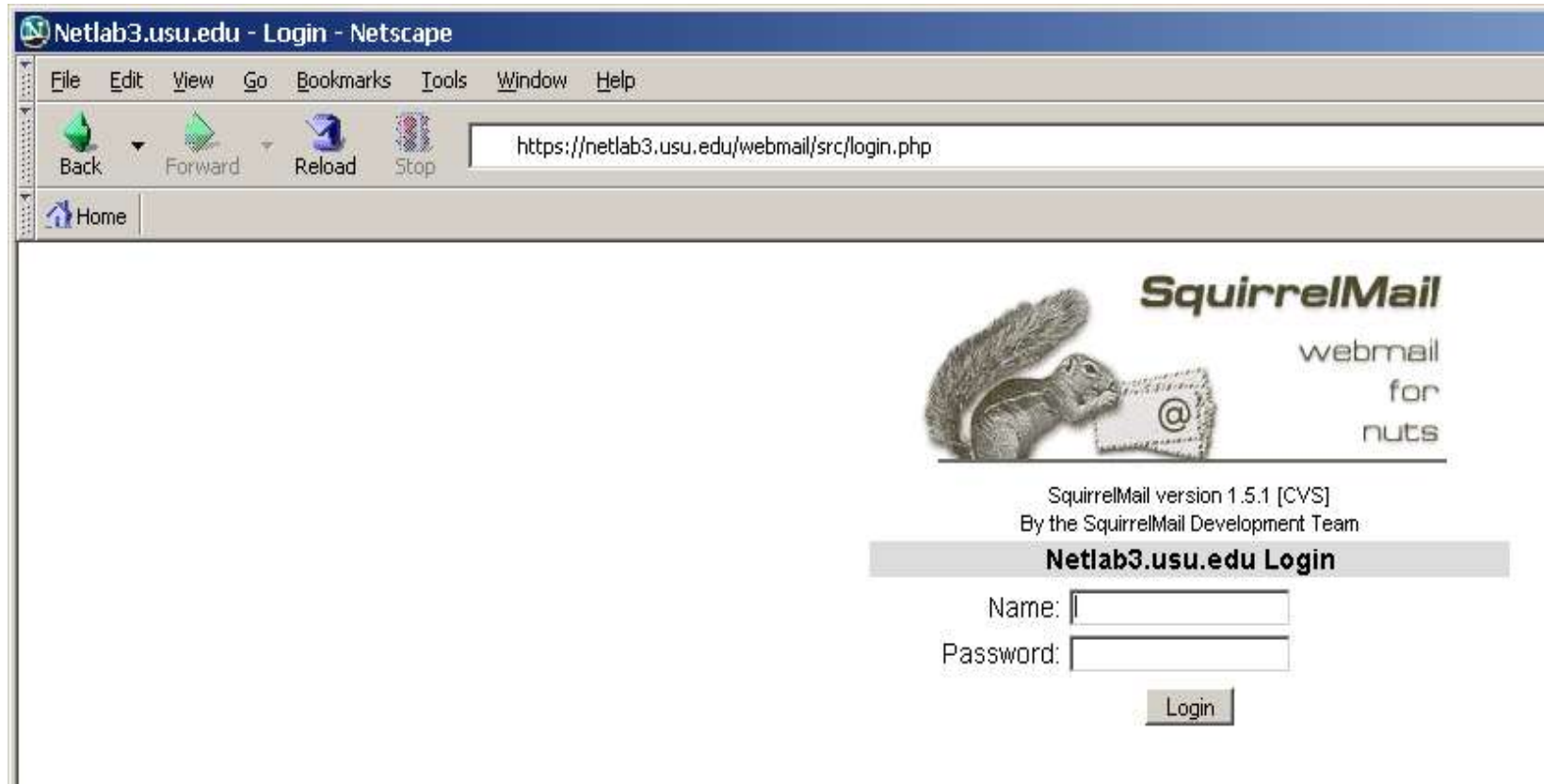
**Add section for Squirrelmail to be run as /webmail**

```
Alias /webmail "/usr/local/squirrelmail"  
<Directory "/usr/local/squirrelmail">  
    Options Indexes MultiViews  
    AllowOverride None  
    Order allow,deny  
    Allow from all  
</Directory>
```

**/usr/local/squirrelmail is outside of document root,  
thus use a <Directory> clause to permit web access.**

**/usr/local/apache2/bin/apachectl restart  
to use new settings**






*And it works! Over https too.*



## *Pause while we check email...*

**Folders**  
Last Refresh: Sat, 2:50 pm  
(refresh folder list)

Save folder tree

 INBOX  
 INBOX.Trash  
 INBOX.Drafts  
 INBOX.Sent  
 Trash

Current Folder: **INBOX**  
[Compose](#) [Addresses](#) [Folders](#) [Options](#) [Search](#) [Help](#) [Calendar](#) [Fetch](#)  
[Sign Out](#) [SquirrelMail](#)

You are using SquirrelMail's web-based email client. If you run into any bugs or have suggestions, please report them to our [mailing list](#)

[Previous] [Next](#) [ 1 2 ] [\[Show All\]](#) Viewing Messages: 1 to 15 (16 total)

[Read](#) [Unread](#) [Forward](#) [Delete](#) ☐ Bypass Trash INBOX [Move](#)

<input type="checkbox"/> From <input type="checkbox"/>	Date ▼	Subject <input type="checkbox"/>
<input type="checkbox"/> root	Mar 28, 2004	<a href="#">SuSEconfig: openssh-changes</a>
<input type="checkbox"/> root	Mar 17, 2004	<a href="#">SuSEconfig: openssl-notify</a>
<input type="checkbox"/> root	Mar 17, 2004	<a href="#">SuSEconfig: openssh-changes</a>
<input type="checkbox"/> root	Feb 15, 2004	<a href="#">SuSEconfig: openssl-notify</a>
<input type="checkbox"/> root	Feb 15, 2004	<a href="#">SuSEconfig: openssh-changes</a>
<input type="checkbox"/> root	Feb 15, 2004	<a href="#">SuSEconfig: SuSEfirewall2 update</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: xntp.caveats</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: SuSEfirewall2 update</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: pine-notify</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: openssl-notify</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: openssh-changes</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: samba-notify</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: ntop-notify</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: bind-sysconfig-named</a>
<input type="checkbox"/> root	Feb 14, 2004	<a href="#">SuSEconfig: mailman-notify</a>

[Previous] [Next](#) [ 1 2 ] [\[Show All\]](#) Viewing Messages: 1 to 15 (16 total)

## *Lessons*

**Major applications are often suitable for building from sources. Benefits can outweigh the effort.**

**Helper apps may be old with little effect and we can install them from RPMs, but see next point**

**Scattering of third party apps suggests building into directory `/usr/local` for our own apps**

**When in doubt examine a source SRPM for building details, then consider it or original sources**

**Locate missing files before overreacting**

**Keep notes, it will be much easier next time**

## *But what about –devel- RPMs?*

**Binary packages, say ldap, were installed but our building an application from source code resulted in not finding ldap.h and similar**

**The default system is binary-only; things are already built (their way) by the vendor**

**Binary RPMs are made from sources by the vendor, and shipped without the building materials**

**Building from local sources requires access to the header files of supporting material, which are in the –devel- RPMs**

# *A hypothetical emergency*

**Suppose a very serious security vulnerability is found in say openssl**

**The vendor has not yet provided a fresh binary RPM, nor a fresh source RPM**

**We feel we must upgrade now, *today***

**What to do?**

**Vendor material is spread over directories different than that of the primary author; there are very likely other nuances and important changes.**

**Try tweaking source RPM steps to use newest source**

## *A typical beware from author*

Versions 0.9.7a, 0.9.7b, and 0.9.7c of OpenSSL are affected by this issue. Any application that makes use of OpenSSL's SSL/TLS library may be affected. Please contact your application vendor for details.

### Recommendations

-----

Upgrade to OpenSSL 0.9.7d or 0.9.6m. Recompile any OpenSSL applications statically linked to OpenSSL libraries.

### Checking our libraries:

**.a=static lib, .so=shared lib**

```
# ls -l /usr/lib/libssl*
-rw-r--r-- 1 root root 305924 Mar  2 10:33 /usr/lib/libssl.a
lrwxrwxrwx 1 root root 11 Feb 14 23:24 /usr/lib/libssl.so -> li
bssl.so.0
lrwxrwxrwx 1 root root 15 Feb 14 22:33 /usr/lib/libssl.so.0 ->
libssl.so.0.9.7
-r-xr-xr-x 1 root root 223898 Mar  2 10:34 /usr/lib/libssl.so.0.9.7
```

### **ldd program** lists shareable libs used by program

```
$ ldd /usr/local/apache2/bin/httpd
libssl.so.0.9.7 => /usr/lib/libssl.so.0.9.7 (0x4002a000)
libcrypto.so.0.9.7 => /usr/lib/libcrypto.so.0.9.7 (0x4005a000)
libaprutil-0.so.0 => /usr/local/apache2/lib/libaprutil-0.so.0
0)
```



## *Plan A*

**Obtain the current app source RPM, unpack it,  
examine its layout and patches**

**SPECS/app.spec file has layout**

**SOURCES/\* has patches and the original sources**

**We replace old with new original sources**

**Patches may no longer apply, if so remove their  
lines from the .spec file**

**Build the new binary RPM from new sources:**

**rpmbuild -bb path/app.spec      create new binary RPM**

**rpm -e oldapps      remove current app RPMs**

**rpm -i path/app.rpm      install our replacement**



# *Portion of application .spec file*

```
Version:      0.9.7b
Release:      133
Summary:      Secure Sockets and Transport Layer Security
URL:          http://www.openssl.org/
Source:       http://www.%(name).org/source/%(name)-%(version).tar.bz2
Source10:     README.SUSE
Patch0:       openssl-0.9.6d.dif
Patch1:       openssl-0.9.6d-flags-priority.dif
Patch7:       openssl-0.9.6c-ppc64.diff
Patch8:       openssl-hppa-config.diff
Patch9:       openssl-0.9.6g-alpha.diff
# http://www-124.ibm.com/developerworks/projects/libica/
Patch10:      openssl-0.9.7b-ICA_engine-070803.patch.bz2
Patch15:      openssl-0.9.7b-asn1.dif
Patch16:      openssl-0.9.7-CAN-2004-0112.dif
Patch17:      openssl-CAN-2004-0079.dif
BuildRoot:    %[_tmppath]/%(name)-%(version)-build
```

**Our new original source is for openssl-0.9.7d**

## Details

```
%prep
%setup -q
%patch -p1
%patch1 -p1
%patch7 -p1
%patch8
%patch9 -p1
%patch10 -p1
%patch15 -p1
%patch16 -p1
%patch17 -p1
```

Openssl.spec detail, for patches.

This applies patches. If failures occur try without them.

## Tarballs

Here you can find all distribution tarballs (and sometimes corresponding patches) of the various O from the OpenSSL FTP area under <ftp://ftp.openssl.org/source/>. Tarballs containing a snapshot are under <ftp://ftp.openssl.org/snapshot/>.

Bytes	Timestamp	Filename
2257721	Mar 17 13:14:47 2004	<a href="#">openssl-engine-0.9.6m.tar.gz</a> (PGP sign)
2798433	Mar 17 13:13:26 2004	<a href="#">openssl-0.9.7d.tar.gz</a> (PGP sign) [LATEST]
2184918	Mar 17 13:11:47 2004	<a href="#">openssl-0.9.6m.tar.gz</a> (PGP sign)
2255800	Nov 4 12:53:07 2003	<a href="#">openssl-engine-0.9.6l.tar.gz</a> (MD5) (PGP sign)
2183726	Nov 4 12:53:03 2003	<a href="#">openssl-0.9.6l.tar.gz</a> (MD5) (PGP sign)
2183608	Sep 30 14:50:16 2003	<a href="#">openssl-0.9.6k.tar.gz</a> (MD5) (PGP sign)
2791797	Sep 30 14:50:15 2003	<a href="#">openssl-0.9.7c.tar.gz</a> (MD5) (PGP sign)

## *The expected details effect*

**Doing `rpmbuild -bb openssl.spec` yields building**

```
+ echo 'Patch #0 (openssl-0.9.6d.dif):'  
Patch #0 (openssl-0.9.6d.dif):  
+ patch -p1 -s  
1 out of 2 hunks FAILED -- saving rejects to file ssl/s2_lib.c.rej  
error: Bad exit status from /var/tmp/rpm-tmp.42499 (%prep)  
  
RPM build errors:  
Bad exit status from /var/tmp/rpm-tmp.42499 (%prep)
```

**Removing patches helped, code built, RPM build failed**

**Consider manual moving built code to system, hope for the best**

**See if patch files exist to update vendor code to newest material**

**Check author's CVS repository for detailed changes, do manually**

## ***Plan B just build a new copy***

**Obtain latest openssl sources from original author**

**Follow building instructions**

**Build into /usr/local**

**Selectively move libraries etc one by one to  
systems space, preserving original files there**

**Be wary of symbolic links, keep them usable**

**Be wary of shared libraries in system but none in  
this generic build (reread INSTALL file for hints)**

**When vendor RPM is ready, undo the copying and  
install the new RPM**

## *Plans A and B*

**With source RPMs the .spec script and patches can do almost anything, treating original source as a malleable object subject to much “improvement”  
Such alchemy is difficult to track and compensate when using unpatched replacement sources**

**Results (plan A) can deviate markedly from the author’s distribution (plan B)**

**By the way, SuSE’s latest openssl had the security patches, but the package name did not match the openssl.org naming scheme. Old base plus patches rather than new base. Still, different results.**

## *A middle ground?*

**Suppose we take a source RPM and modify the selection of features of an application**

**Then we rebuild and install it (oops, overwrites old)**

**We get wanted features while building on top of vendor's work to configure the application**

**The RPM shows in the list of vendor supplied units and is subject to replacement while upgrading, unless we wisely re-label it**

**Dependencies from new features may not appear in the new RPM, unless we add them**

**We must be very careful about vendor patches**

## Summary

**Building applications from original sources may require building selected worker apps from sources and storing them in /usr/local (avoid editing *configure*, feature changes)**

**Various –devel- RPMs may be required to supply needed files from vendor built workers**

**Vendor built programs go into system areas**

**Our built programs should go into local areas**

**This is a form of “federation” (non-interfering co-existence)**

## *Summary*

**Some applications are written in such a manner as to not work correctly on the current operating system**

**Vendors often supply corrective patches to help.  
Review those patches if you build from original sources**

**Keep in mind that term “Unix” is an oxymoron**



## ***For fun, unencumber NTP***

**Fetch ntpd-4.2.0-tar.gz from ftp.udel.edu, pub/ntp**

**Unpack within /home/ntp**

**./configure (defaults to /usr/local area)**

**make**

**make install**

**Shutdown Linux vendor edition (/etc/init.d area)**

**/usr/local/bin/ntpd -g to start the daemon**

**(uses /etc/ntp.conf and /etc/ntp.drift)**

**Create new ntpd startup script in /etc/init.d (see Notes)**

**rpm -e xntpd-x.x.x or hide old executables**

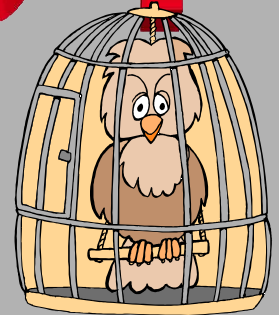
## *We have become*

<b>Job:</b>	<b>Builder</b>
<b>Hobbies:</b>	<b>Building</b>
<b>Favourite song:</b>	<b>Can we fix it</b>
<b>Favourite website:</b>	<b>www.bobthebuilder.com</b>
<b>Likes:</b>	<b>Building, time with my Friends</b>
<b>Dislikes:</b>	<b>Foxes, I leave all that to Wendy</b>
<b>Best friend:</b>	<b>Everybody</b>
<b>Ambition:</b>	<b>Fix everything!</b>

**Bob**

A cartoon illustration of Bob the Builder, a man with a yellow hard hat, a red and yellow checkered shirt, and blue overalls. He is holding a rolled-up blueprint in his right hand and a paint bucket with a brush in his left hand. He is standing on a blue background with white clouds.

# Questions?





MindWorks Inc. Ltd  
210 Burnley Road  
Weir  
Bacup  
OL13 8QE UK

Telephone: +44 (0) 170 687 1900

Fax: +44 (0) 170 687 8203

Web: [www.mindworksuk.com](http://www.mindworksuk.com)

Email: [training@mindworksuk.com](mailto:training@mindworksuk.com)