Compile Nmap for Android

This tutorial will show you how to compile the latest version of Nmap for your Android device starting with a standard Ubuntu install. I will offer instructions on how to obtain two versions of compiler that I've had success compiling software for Android. I will show the Android NDK and the free Lite ARM compiler from Mentor (formally Code Sorcery). Hopefully you can take this instruction to try and compile other tools for Android.

The build environment and instructions come from an auditor with strong technical skills but somebody who is not a programmer or developer so hopefully my view point can help other individuals who are also not developers. I've built cross-compile environments for Openwrt, Nokia Maemo, Familiar Linux (iPaq) in the past but always from piecing together instructions from multiple Google queues and forum searches. I'm creating this document so it will be helpful for someone somebody elses Google search.

After the Ubuntu installation here are ALL the steps you can/should take to compile Nmap for Android. I like vim as my command-line editor. You can use which ever editor you prefer.

Here is a quick rundown of what is done. Everything (almost) is done from a terminal window.

1. I update all Ubuntu software and install all files and tools to compile software on Ubuntu
2. I download the software required to compile for Android
3. Setup the environment to compile for Android
4. I create a source folder in the home directory for downloading and compiling the software.
5. Download the software, patch, configure, and compile.
6. Install Android SDK Platform Tools to copy files to your phone
7. Copy files to the phone and set PATH environment variable.

```
$sudo -s
#apt-get update
#apt-get upgrade
#apt-get install vim build-essential
#cd /usr/local
#wget
https://sourcery.mentor.com/sgpp/lite/arm/portal/package9728/public/ar
m-none-linux-gnueabi/arm-2011.09-70-arm-none-linux-gnueabi-i686-pc-
linux-gnu.tar.bz2
#tar jxf arm-2011.09-70-arm-none-linux-gnueabi-i686-pc-linux-
gnu.tar.bz2
#rm arm-2011.09-70-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
#exit
$
```

Ubuntu is now current with the tools required to compile for Android. Now to create a simple shell script to setup the environment variables for cross compiling.

```
$mkdir ~/source
```

```
$cd ~/source
$vim setenv.sh
```

Include the following information in setenv.sh. This script will set the environment to allow you to use the Android / ARM compiler instead of the native compiler.

```
#!/bin/bash
export ac_cv_linux_vers=2
export CC=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-gcc
export GCC=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-gcc
export CXX=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-g++
export CPP=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-cpp
export LD=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-ld
export AR=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-ar
export AS=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-as
export NM=/usr/local/arm-2011.09/bin/arm-none-linux-gnueabi-nm
export RANLIB=/usr/local/arm-2011.09/arm-none-linux-gnueabi/bin/ranlib
export CC1=/usr/local/arm-2011.09/libexec/gcc/arm-none-linux-gnueabi/4.6.1/cc1
export PATH=/usr/local/arm-2011.09/bin:/usr/local/arm-2011.09/:/usr/local/arm-
2011.09/lib:/usr/local/arm-2011.09/libexec/gcc/arm-none-linux-gnueabi/4.6.1:$PATH
export ac_cv_func_getpgrp_void=yes
export ac_cv_func_setpgrp_void=yes
export LDFLAGS="-static"
export CFLAGS="-Os -s"
```

```
$source setenv.sh
$wget http://nmap.org/dist/nmap-5.61TEST4.tar.bz2
$tar jxf nmap-5.61TEST4.tar.bz2
$cd nmap-5.61TEST4
```

At this point you can configure and compile just Nmap. This will be without LUA library support (--without-liblua) which will disable all of the NSE scripts that have been created for Nmap as well as the additional command line tools ncat, ndiff, and nping.

```
$ ./configure --host=arm-linux-androideabi --without-zenmap --without-
liblua --without-nping --without-ndiff --without-ncat --with-
libpcap=internal --with-pcap=linux --enable-static --prefix=/data/opt
```

If you want to include these scripts and tools you will have to modify the Nmap source code. If you try to compile with LUA support you will get this error.

```
strict-aliasing   -DLUA_USE_POSIX -DLUA_USE_DLOPEN   -c -o llex.o llex.c
llex.c: In function 'trydecpoint':
llex.c:181: error: 'struct lconv' has no member named 'decimal_point'
make[1]: *** [llex.o] Error 1
make[1]: Leaving directory `/home/edge/source/nmap-5.51/liblua'
make: *** [lua_build] Error 2
```

If you try to compile without LUA but want nping you get this error.

```
In file included from ArgParser.cc:94:
nping.h:116:26: error: sysexits.h: No such file or directory
make[3]: *** [ArgParser.o] Error 1
make[3]: Leaving directory `/home/edge/source/nmap-5.51/nping'
make[2]: *** [all] Error 2
make[2]: Leaving directory `/home/edge/source/nmap-5.51/nping'
make[1]: *** [build-nping] Error 2
make[1]: Leaving directory `/home/edge/source/nmap-5.51'
make: *** [all] Error 2
```

Vlatko Kosturjak <kost () linux hr> comes to the rescue with his patch that allows you to compile with LUA and nping. I'm hosting the patch from my site just in case it is removed from the forums where he posted it.

```
$cd ~/source/ nmap-5.61TEST4
$mkdir android
$wget http://www.jedge.com/code/nmap.android.patches.diff -O android/
nmap.android.patches.diff
$patch -N -p1 < android/nmap.android.patches.diff
$ ./configure --host=arm-linux-androideabi --without-zenmap --with-
liblua=included --with-libpcap=internal --with-pcap=linux --enable-
static --prefix=/data/opt
$make
$sudo make install
```

One my phone I created the directory opt in the data directory where I install all of my tools (/data/opt). That is why you see --prefix set to /data/opt. You can set it to whatever directory you want but remember I use this directory throughout my instructions. When you "install" nmap on your Linux host system it will be placed in /data/opt/bin and /data/opt/share. I mirror these same directories on my Android phone.

I believe you can run nmap on a phone that has not been rooted by sticking the files in /data/data/<terminal_app_directory>/bin but I will not be discussing how to accomplish that or what pitfalls exist. My phone is already rooted and yours should be too if you want to be successful with this tutorial. Rooting your phone is outside the scope of this tutorial.

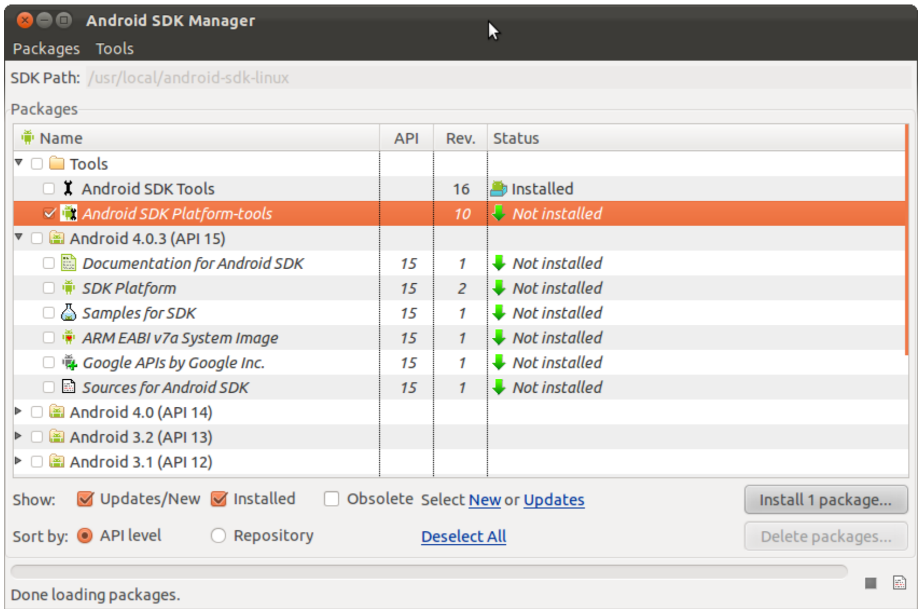
Install Android SDK and pushing files to the phone

One of the best ways to get files copied to your Android phone is to use the command 'adb' which is part of the platform-tools of the Android SDK. Below are the steps to obtain the SDK and use 'adb' to push the compiled nmap files to your phone.

```
$sudo -s
#apt-get install openjdk-7-jdk
#cd /usr/local
# wget http://dl.google.com/android/android-sdk_r16-linux.tgz
#tar android-sdk_r16-linux.tgz
```

This next step is the only part where you need a GUI. Running /usr/local/android-sdk-linux/tools/android will open up a window where you need to select Android SDK Platform Tools, deselect Android 4.0.x, and click Install 1 package... (See Screenshot below).

```
#android-sdk-linux/tools/android &
```



Android SDK Manager

SDK Path: /usr/local/android-sdk-linux

Packages

Name	API	Rev.	Status
▼ ☐ Tools			
☐ Android SDK Tools		16	Installed
✓ Android SDK Platform-tools		10	Not installed
▼ ☐ Android 4.0.3 (API 15)			
☐ Documentation for Android SDK	15	1	Not installed
☐ SDK Platform	15	2	Not installed
☐ Samples for SDK	15	1	Not installed
☐ ARM EABI v7a System Image	15	1	Not installed
☐ Google APIs by Google Inc.	15	1	Not installed
☐ Sources for Android SDK	15	1	Not installed
▶ ☐ Android 4.0 (API 14)			
▶ ☐ Android 3.2 (API 13)			
▶ ☐ Android 3.1 (API 12)			

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Done loading packages.

```
#export PATH=/usr/local/android-sdk-linux/platform-tools:$PATH
#adb devices
```

Output examples

```
    List of devices attached
    ????????????    offline
```

Unplug your phone and plug it back in. You should see the following output.

```
    List of devices attached
    364247A74CE500FD        device
```

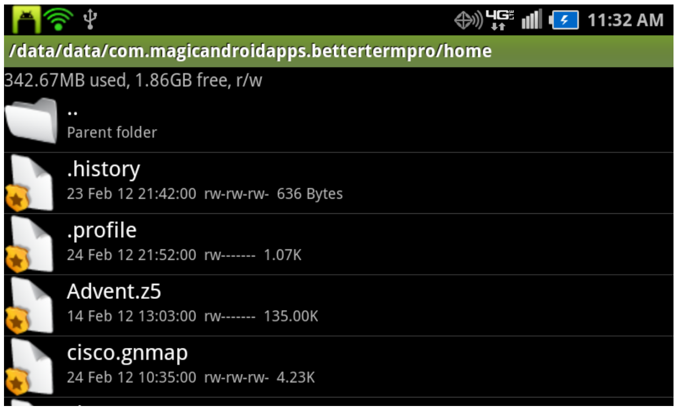
```
#adb remount
#adb shell mkdir /data/opt
#adb chmod 755 /data/opt
#adb push /data/opt /data/opt
```

This will push the bin and share directories created on Ubuntu that were installed when you ran 'make install' from the Nmap directory. They will be pushed to the corresponding directory on the phone.

Running Nmap on your phone

I use BTEP (Better Terminal Emulator Pro) as my terminal of choice on my Android phone. The home directory is located in /data/data/com.magicandroidapps.bettertermpro/home
I also use Root Explorer to traverse the file structure and open up files for editing.

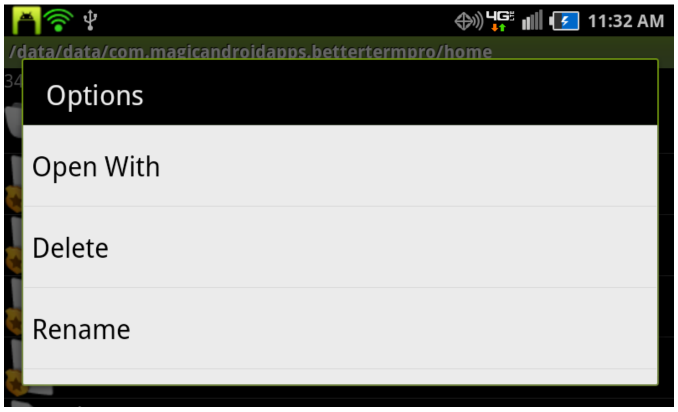
All applications I compile for Android I place in my /data/opt directory and I modify .profile of BTEP to include the executable directory in my PATH (/data/opt/bin). I also include /data/opt/lib in LD_LIBRARY_PATH.


/data/data/com.magicandroidapps.bettertermpro/home

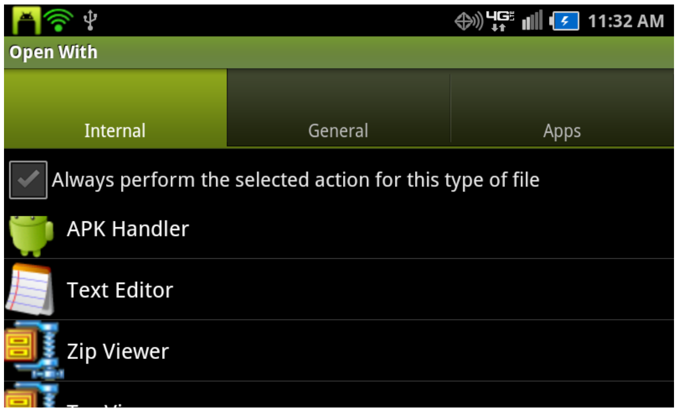
342.67MB used, 1.86GB free, r/w

..	Parent folder
.history	23 Feb 12 21:42:00 rw-rw-rw- 636 Bytes
.profile	24 Feb 12 21:52:00 rw------- 1.07K
Advent.z5	14 Feb 12 13:03:00 rw------- 135.00K
cisco.gnmap	24 Feb 12 10:35:00 rw-rw-rw- 4.23K

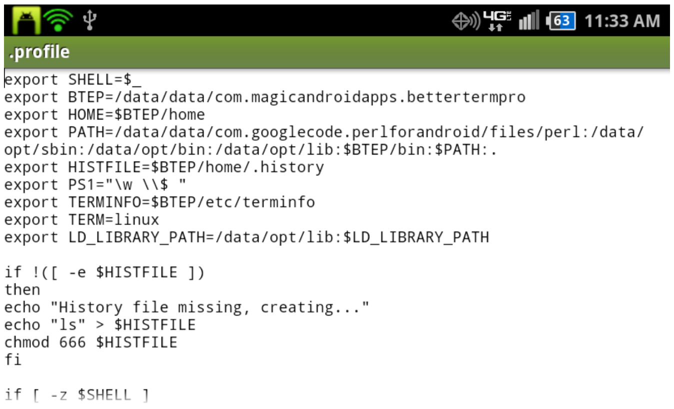
Open Root Explorer and travers the directories until you get to the home directory for BTEP.



Select .profile and hold until the Options show up. You will choose Open With



Choose the Text Editor to open .profile so we can modify the file.

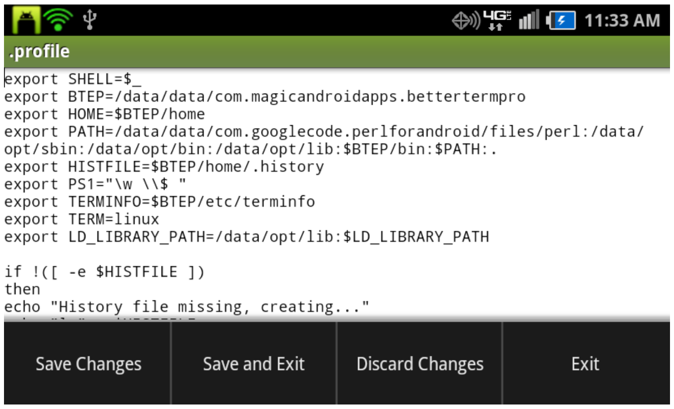


```
export SHELL=$_
export BTEP=/data/data/com.magicandroidapps.bettertermpro
export HOME=$BTEP/home
export PATH=/data/data/com.googlecode.perlforandroid/files/perl:/data/
opt/sbin:/data/opt/bin:/data/opt/lib:$BTEP/bin:$PATH:.
export HISTFILE=$BTEP/home/.history
export PS1="\w \\$ "
export TERMINFO=$BTEP/etc/terminfo
export TERM=linux
export LD_LIBRARY_PATH=/data/opt/lib:$LD_LIBRARY_PATH

if !([ -e $HISTFILE ])
then
echo "History file missing, creating..."
echo "ls" > $HISTFILE
chmod 666 $HISTFILE
fi

if [ -z $SHELL ]
```

Modify the PATH variable and add the export LD_LIBRARY_PATH statement.



```
export SHELL=$_
export BTEP=/data/data/com.magicandroidapps.bettertermpro
export HOME=$BTEP/home
export PATH=/data/data/com.googlecode.perlforandroid/files/perl:/data/
opt/sbin:/data/opt/bin:/data/opt/lib:$BTEP/bin:$PATH:.
export HISTFILE=$BTEP/home/.history
export PS1="\w \\$ "
export TERMINFO=$BTEP/etc/terminfo
export TERM=linux
export LD_LIBRARY_PATH=/data/opt/lib:$LD_LIBRARY_PATH

if !([ -e $HISTFILE ])
then
echo "History file missing, creating..."
```

Select your menu button on your phone and choose Save and Exit

To have your changes take effect you can close and reopen BTEP or run #source .profile from the command line.

You can now run Nmap from your Android phone. A program you successfully cross compiled yourself. These steps can be modified to compile other software for your phone. Your mileage will vary with other software. Especially since Android uses a stripped down libc called Bionic which will prevent software from compiling or running correctly.

Special Thanks to Vlatko Kosturjak <kost () linux hr> http://k0st.wordpress.com/ who got nmap to compile with liblua support as well as nping to compile.

https://secwiki.org/w/Nmap/Android
http://seclists.org/nmap-dev/2012/q1/135
http://seclists.org/nmap-dev/2010/q2/1021
http://k0st.wordpress.com/2012/01/12/nmap-5-61test4-on-android/