Installation of TOMOYO, IMA, the TPM emulator and TrouSerS on Android

Stephan Heuser
stephan.heuser@sit.fraunhofer.de

May 19, 2010

This document describes how to build and start an Android Image which uses TOMOYO (Mandatory Access Control) [3], IBM IMA (Integrity Measurement) [1, 6], the TPM emulator [2, 7] and TrouSerS (TCG Software Stack) [5]. This guide is based on the TOMOYO Installation manual [4].

1 Requirements

- x86-Linux based System
- Sun Java 5 JDK, JRE

  For Ubuntu: Java 5 was removed from recent Ubuntu Versions. Either install it manually or use the older Packages from Dapper/Jaunty. Add:

  ```
  deb http://de.archive.ubuntu.com/ubuntu/ dapper main
  deb http://de.archive.ubuntu.com/ubuntu/ jaunty multiverse
  deb http://de.archive.ubuntu.com/ubuntu/ jaunty-updates \
  multiverse
  ```

  to `/etc/apt/sources.list`. Update the package index and install Java 5:

  ```
  apt-get update
  apt-get install sun-java5-bin sun-java5-jre sun-java5-jdk
  ```

- Android repo Script (`http://source.android.com/download`)
- droid-wrapper (`http://github.com/tmurakam/droid-wrapper`)

2 Building and Running

2.1 Kernel

In order to build a complete Android OS image you first need to build the Linux kernel for the emulator.

2.1.1 Download the Android sourcecode and build environment.

To build the kernel a toolchain is needed. A complete, prebuilt toolchain is available from Google. The entire procedure to download and build the SDK, which contains the prebuilt toolchain, is described at `http://source.android.com/download`. Do not build anything yet, we just need the toolchain for now. I will refer to the directory the Android sourcecode was extracted to as `ANDROID_SOURCE` from now on.

2.1.2 Download, Patch and Compile the Android goldfish kernel

The goldfish kernel is the Linux kernel used for the Android emulator.

1. The kernel sourcecode is available at `http://android.git.kernel.org/?p=kernel/common.git;a=summary`. Download the current 2.6.29 snapshot (`android-goldfish-2.6.29`) from the repository.

2. Extract the kernel:

   ```
   tar -xzf common-refs_heads_android-goldfish-2.6.29 \
   .tar.gz
   ```

3. Move to the extracted kernel directory and remove world writable permissions from the source code:

   `cd common` (i will refer to this directory as GOLDFISH from now on)

   ```
   find -print0 | xargs -0 chmod go-w --
   ```

4. Download and extract the TOMOYO patchset. Do not apply any of the patches yet!

   ```
   wget http://osdn.dl.sourceforge.jp/tomoyo/30297/ \
   ccs-patch-1.6.8-20090528.tar.gz

   tar -zxf ccs-patch-1.6.8-20090528.tar.gz
   ```

5. Build the Kernel using the previously downloaded Android build environment.

```
ARCH=arm CROSS_COMPILE=ANDROID_SOURCE/prebuilt/ \
linux-x86/toolchain/arm-eabi-4.2.1/bin/arm-eabi- \
make goldfish_defconfig

ARCH=arm CROSS_COMPILE=ANDROID_SOURCE/prebuilt/ \
linux-x86/toolchain/arm-eabi-4.2.1/bin/arm-eabi- \
make -s -jN
```

   You can specify the number of threads to use by setting the parameter -jN. N is the number of threads (e.g. on a dual core system one might use `-j2`).

6. Apply the kernel patches available at `http://www.vogue-project.de/cms/front_content.php?idcat=10`. Patches may fail since the Goldfish sourcecode changes over time. In case of failing patches either use quilt for automated patch managment or analyste the rejected patches and apply the changes yourself. The patchset consists of five patches.

   `patch -p1 < 001_ccs-patch-2.6.29.patch` (TOMOYO for Android)

   `patch -p1 < 002_ibm_ima_2.6.29.1.patch` (IBM IMA)

   `patch -p1 < 003_ima_patch` (IBM IMA fixes for Android)

   `patch -p1 < 004_tpmd.patch` (TPM emulator device)

   `patch -p1 < 005_config.patch` (Kernel configuration files)

   You can test the patches using the `--dry-run` parameter before applying them.

7. Build the kernel again.

```
ARCH=arm CROSS_COMPILE=ANDROID_SOURCE/prebuilt/ \
linux-x86/toolchain/arm-eabi-4.2.1/bin/arm-eabi- \
make -s -jN
```

   (where N is the number of threads to use)

2.1.3 Cross-compiling the GMP library for Android

This guide is mainly based on the droid-wrapper (`http://github.com/tmurakam/droid-wrapper`) to cross-compile the GMP library (`http://gmplib.org`) for Android.

1. Build the Android source code

   Follow the instructions at `http://source.android.com/download` to build the Android source code.

2. Install ruby

   Install ruby if it is not already installed. It is required by the droid-wrapper, On Ubuntu/Debian-based Systems:

   ```
   sudo apt-get install ruby
   ```

3. Download and install droid-wrapper

   (a) Download the droid-wrapper script from `http://github.com/tmurakam/droid-wrapper`.

   (b) Extract the droid-wrapper

   (c) Move to the extracted droid-wrapper source directory

   ```
   sudo make install
   ```

   This will install `droid-gcc`, `droid-g++`, and `droid-ld` under `/usr/local/bin`. There might be following two errors, which can be ignored:

   ```
   /bin/rm:  cannot remove ‘/usr/local/bin/droid-ld’:  No such
   file or directory
   ```

   ```
   /bin/rm:  cannot remove ‘/usr/local/bin/droid-g++’:  No such
   file or directory
   ```

4. Set environment variables for droid-wrapper

   Specify following environment variables before using the droid-wrapper:

   ```
   export DROID_ROOT=path/to/ANDROID_SOURCE
   ```

   Make sure that you set the `ANDROID_SOURCE` path correctly.

```
export DROID_TARGET="generic"
```

5. Download and cross-compile the GMP library

   The GNU Multiple Precision Arithmetic Library source code is available at `http://gmplib.org`. Download the version 4.3.2 (`gmp-4.3.2.tar.bz2`).

6. Extract the gmp-4.3.2.tar.bz2 library

   ```
   tar -xjvf gmp-4.3.2.tar.bz2
   ```

7. Move to the extracted directory

   ```
   mkdir INSTALL
   ```

   ```
   CC=droid-gcc LD=droid-ld ./configure -prefix=path/to/INSTALL \
   -build=i686-pc-linux-gnu -host=arm-linux-gnueabi
   ```

   Make sure that you set the –prefix option (path to the `INSTALL` directory) correctly.

   ```
   make
   ```

   ```
   make install
   ```

   The generated static and shared library can be found under `INSTALL/lib`. The header file (gmp.h) can be found under `INSTALL/include`.

2.1.4 Patch and compile Android

The TPM emulator, TPM tools and TrouSerS have to be added to the Android source before building it.

1. Change to the Android directory. Perform cleanup of the build directory.

   ```
   cd ANDROID_SOURCE
   ```

   ```
   make clean
   ```

2. Download `mydroid.patch` from `http://www.vogue-project.de/cms/front_content.php?idcat=10`

3. Apply the patch.

   ```
   patch -p1 < mydroid.patch
   ```

4. Copy the previously built `libgmp.a` to `ANDROID_SOURCE/frameworks/base/libs/libgmp`

5. Build Android.

   `make -jN` (where N is the number of threads to use).

2.1.5 Preparing the Image, adding ccs-tools

The TOMOYO ccs-tools have to be added to the previously built Android image.

1. `ANDROID_SOURCE/out/target/product/generic` contains the Android image files. Copy the files `system.img`, `ramdisk.img` and `userdata.img` to a new directory, which i will refer to from now on as `ANDROID_IMAGE`. Copy the new kernel image `(GOLDFISH/arch/arm/boot/zImage)`, to this directory and rename it to `kernel.img`

2. Create an sdcard image inside the directory:

   ```
   mksdcard -l Mysdcard 1024M sdcard.img
   ```

   Replace 1024M with the desired size.

3. Add the precompiled ccs-tools to the ramdisk.

   ```
   mkdir initramfs

   cd initramfs

   zcat ../ramdisk.img | cpio -id

   cd sbin

   wget http://tomoyo-android.googlecode.com/files/ \
   ccstools-embedded_bin.tgz

   tar -xzf ccstools-embedded_bin.tgz

   rm ccstools-embedded_bin.tgz

   cd ..

   find .  -print0 | cpio -o0 -H newc | gzip -9 > \
   ../ramdisk.img
   ```

2.1.6 Building ccs-tools on the host system

Download source code from TOMOYO project from SourceForge.jp, extract it and compile it:

```
wget http://sourceforge.jp/frs/redir.php?f=/tomoyo/30298/ \
ccs-tools-1.6.8-20100115.tar.gz

tar -zxf ccs-tools-1.6.8-20100115.tar.gz

cd ccstools

make

make install
```

2.1.7 Preparing and starting the emulator

1. Set the environment variables needed to start the emulator.

   ```
   export EMULATORDIR=ANDROID_SOURCE/out/host/linux-x86/bin
   export SYSTEM=ANDROID_IMAGE
   ```

2. Start the emulator.

   ```
   emulator -kernel $SYSTEM/kernel.img \
   -system $SYSTEM \
   -ramdisk $SYSTEM/ramdisk.img \
   -data $SYSTEM/userdata.img \
   -sdcard $SYSTEM/sdcard.img
   ```

3. Forward Port 7000 to to be able to control TOMOYO externally:

   ```
   adb forward tcp:7000 tcp:7000
   ```

4. Open a shell:

   ```
   adb shell
   ```

5. Create the directory `/data/usr/lib` for tcsd:

   ```
   mkdir /data/usr

   mkdir /data/usr/lib
   ```

6. Start the TOMOYO Policy Editing Agent if it is not already running:

   ```
   ccs-editpolicy-agent 0.0.0.0:7000 &
   ```

7. Start tpmd if it is not already running. The TPM daemon should start during initialization. It fails to start if the socket `/data/tpmd_socket:0` already exists. If this is the case remove the socket file and start tpmd manually. This happens anytime the emulator is killed.

   ```
   rm /data/tpmd_socket:0
   ```

   ```
   tpmd
   ```

   The parameters `-d -f` can be used to retrieve additional debug information and to force tpmd to run in foreground.

8. Start tcsd:

   ```
   tcsd
   ```

   The parameter `-f` may be used to force tcsd to stay in foreground.

9. Initialize the TOMOYO Linux policy:

   ```
   init_policy_android.sh
   ```

10. You should be able to edit TOMOYO policies now from the host system. On the host system:

    ```
    ccs-editpolicy 127.0.0.1:7000
    ```

11. Mount securityfs if it is not already mounted:

    ```
    mount -t securityfs securityfs /sys/kernel/security
    ```

12. Verify that IMA works correctly.

    `/sys/kernel/security/ima/ascii_runtime_measurements` should contain the hash values of apk files and executables if IMA works correctly.

References

[1] Integrity Measurement Architecture (IMA). `http://sourceforge.net/projects/linux-ima/`.

[2] Software-based TPM Emulator. `http://tpm-emulator.berlios.de/`.

[3] Tomoyo Linux. `http://tomoyo.sourceforge.jp/`.

[4] Tomoyo on Android. `http://code.google.com/p/tomoyo-android/wiki/HowToApplyTomoyoOnAndroidARM`.

[5] TrouSerS. `http://trousers.sourceforge.net/`.

[6] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA August 9-13, 2004*, pages 223–238, Berkeley, CA, USA, 2004. USENIX Association.

[7] M. Strasser and H. Stamer. A Software-Based Trusted Platform Module Emulator. In *Trusted Computing - Challenges and Applications. First International Conference on Trusted Computing and Trust in Information Technologies, TRUST 2008 Villach, Austria, March 11-12, 2008*, volume 4968 of *LNCS*, pages 33–47, Berlin, Germany, 2008. Springer.