Secure Middleware for Embedded Systems

Android Compilation and Installation

Lena Voytek

7/24/2020

Contents

List of Figures

Introduction

The main board that the SMES project focuses on is the ASUS TinkerBoard (S). It contains a Rockchip RK3288 SoC with an ARM processor, 2GB of ram, and a Mali GPU in the form factor of a Raspberry Pi 3. ASUS has a custom set of Android builds on their website that work on the TinkerBoard and TinkerBoard S. However, this document describes how to build a custom version of this Android image from source. This allows for editing of the fundamentals of the OS, including the TEE, GPIO connection, and rooting by default. Official edits to the Android source code will be added to a repository located at TBD.

Warning: This process requires around 120GB of free space to run and will end up compressing down to around 60GB in the end.

1 Download Source

The build process has been tested and works properly on Ubuntu 18.04 and Ubuntu 20.04. A Linux device or virtual machine will be necessary for this process.

Begin by opening a terminal and installing the necessary libraries for downloading and compiling the source code. Then setup ccache to speed up the compilation process further. The following can be run on Ubuntu.

```
sudo apt-get install -y openjdk-8-jdk git-core gnupg flex bison gperf build-essential zip
    curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-
    core-dev libx11-dev lib32z-dev libgl1-mesa-dev libxml2-utils xsltproc unzip python
    lzop libncurses5
sudo apt-get install -y ccache
sudo /usr/sbin/update-ccache-symlinks
echo 'export PATH="/usr/lib/ccache:$PATH"' | tee -a ~/.bashrc
source ~/.bashrc
which g++ gcc
```

All necessary command line tools should now be installed and gcc should be using ccache. If the 'which' command gave an output of **/usr/lib/ccache/g++** and **/usr/lib/ccache/gcc** then it is good to go. Also be sure to set up a name and email for git if not done already.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Now install the repo command from Google with the following commands.

```
mkdir ~/bin
sudo curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
echo 'export PATH="~/bin:$PATH"' | tee -a ~/.bashrc
source ~/.bashrc
```

Now create a folder for the source to be stored in and initialize the repository there. Again note that this will take up a lot of space. In this example the folder name is AndroidBuild but can be named whatever. Also the -j4 should be replaced with however many threads

should be dedicated to this setup (e.g. -j16 for a 16 thread cpu). The repo sync will take a while (sometimes several hours) especially on a system with a low threadcount or in a place with slow internet.

```
mkdir AndroidBuild
cd AndroidBuild/
repo init -u https://git@bitbucket.org/TinkerBoard_Android/manifest.git -b sbc/
    tinkerboard/asus/Android-7.1.2
repo sync -j4 -c
```

Once this process completes the Android source tree will be fully available to manipulate and build into a custom image.

2 Build Android

With the Android source repo downloaded, the build process can now begin. In the terminal, navigate to the repository folder that was created earlier and run the following. This will provide some tools specific to using the Tinkerboard's processor.

```
source build/envsetup.sh
lunch rk3288-userdebug
```

Now it is time to build the kernel, this usually takes around 8 minutes using -j8.

```
cd kernel/
make ARCH=arm rockchip_defconfig
make ARCH=arm rk3288-miniarm.img -j8
```

Now build the u-boot binaries, this usually takes around 1.5 minutes with -j8.

```
cd ../u-boot
make rk3288_secure_defconfig
make -j8
```

Next, compile the system image. This usually takes around a few hours at -j8, or 15 minutes on -j32 with an AMD 3950 32 thread processor. If it fails at any point try decreasing the threadcount and restarting the make command.

```
cd ..
export LC_ALL=C
make -j8
```

Finally, transform these images into one single update.img file with the following commands.

```
./mkimage.sh
cd RKTools/linux/Linux_Pack_Firmware/rockdev
./collectImages.sh
./mkupdate.sh
```

The compressed single image file is now available in the current directory and is named update.img. However, this is not quite the final product as this must be transformed into

a .bin for flashing. This will require an obscure totally not sketchy Windows application to complete, as seen in the next section.

3 Extract Image

In order to extract the flashable binary from the update.img file, an application is needed. It can either be run on the Liunux system currently in use with WINE and PlayOnLinux, or update.img can be moved to a Windows 10 system and extracted directly there. Both ways are detailed below.

For both operating systems, the executable can be found at https://github.com/rockchip-linux/tools/blob/master/windows/SpiImageTools_v1.36.zip. Download and extract the zip file to obtain SpiImageTools.exe.

3.1 Extracting on Linux

In order to continue the process on the current Linux system, PlayOnLinux and WINE need to be installed. Run the following commands to do so.

```
sudo apt-get install wine64 playonlinux xterm
```

Now search for PlayOnLinux in the application search bar and open it. The window should look something like the following:

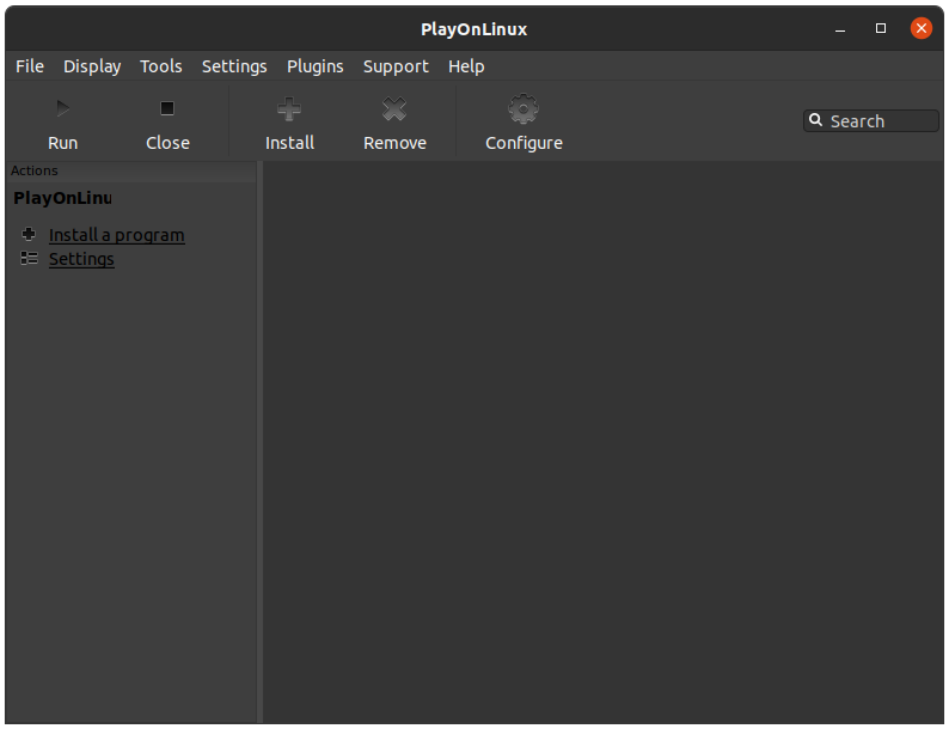

Figure 1: PlayOnLinux Home Screen

Click on the Install button, then click "Install a non-listed program."

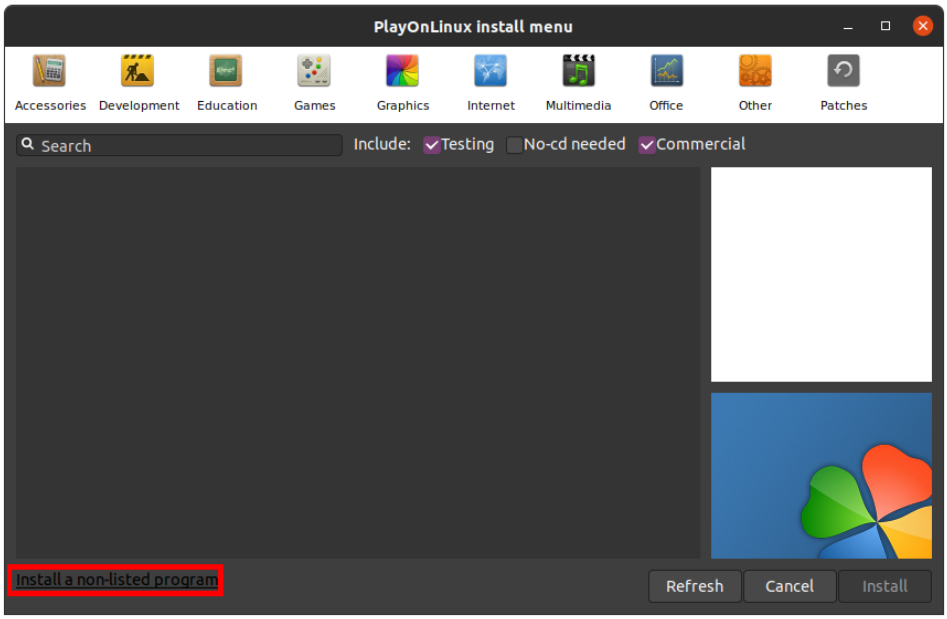

Figure 2: PlayOnLinux Install Screen

Click next until the following menu shows up, and select "Install a program in a new virtual drive."

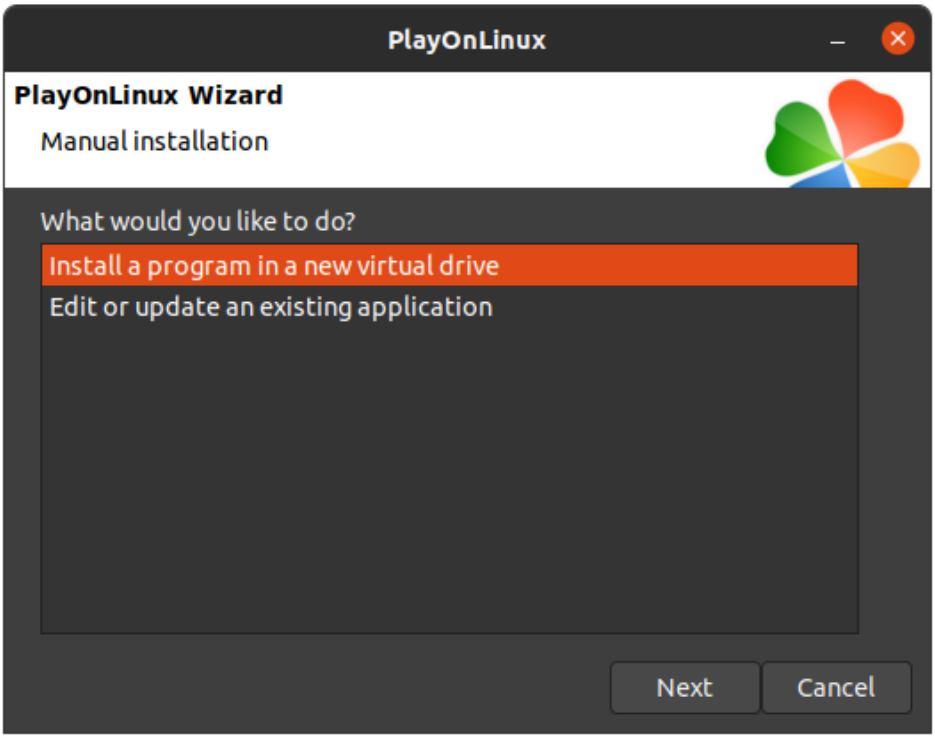

Figure 3: PlayOnLinux Installation Wizard Screen

In the next screen, the virtual drive can be given any name. After that, when it askes what to do prior to installation, leave all boxes unchecked. Finally, select a 64-bit Windows installation and let PlayOnLinux configure Wine accordingly. This should lead to the following window.

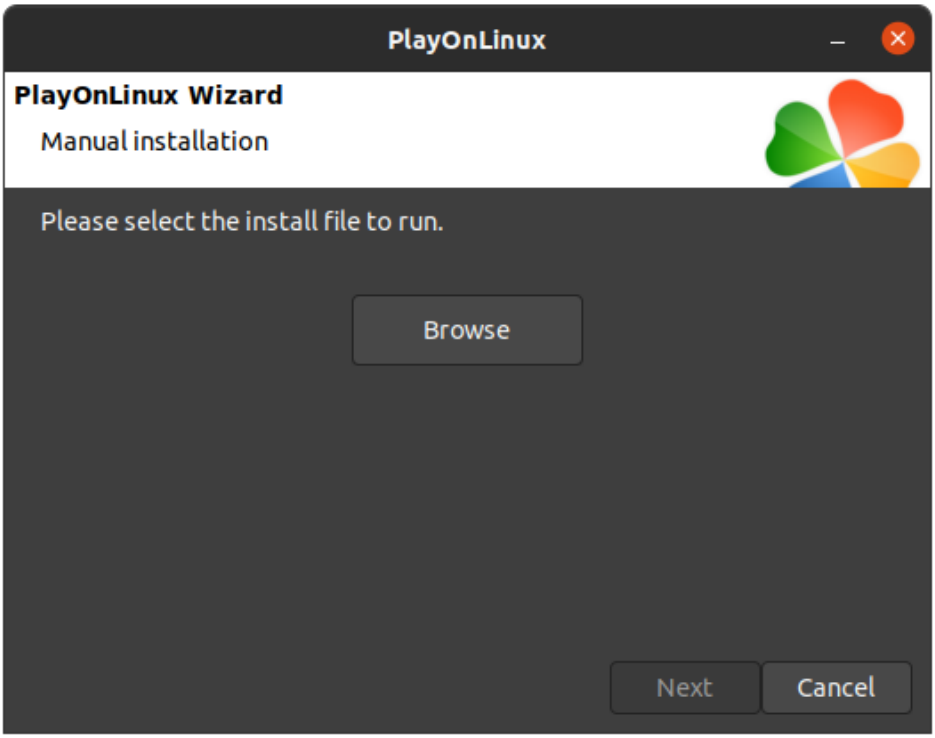

Figure 4: PlayOnLinux Installation Program Finder Screen

Click browse then navigate to the SpiImageTools.exe file that was downloaded earlier, then click next. PlayOnLinux will setup its environment then run the application automatically. When it opens it should look like the following.

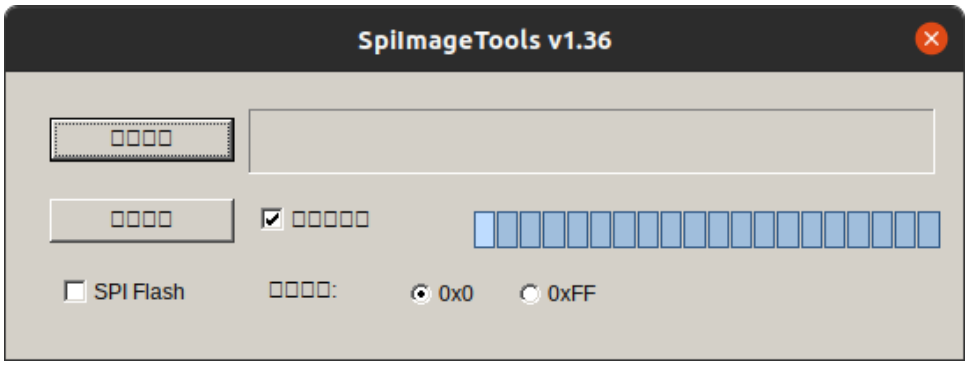

Figure 5: The cryptic SpiImageTools program

All the text will probably show up as tofu boxes. This is fine. It should still wortk as intended. Click the top button to open a file dialog box. Navigate to the update.img located in repo directory/RKTools/linux/Linux_Pack_Firmware/rockdev. This will have to be found through The / directory, most likely followed by home/username. Once the img file is uploaded, click the bottom button and the blue bar will start pinging back and forth for a little while. The process will be complete when the following dialog box pops up.

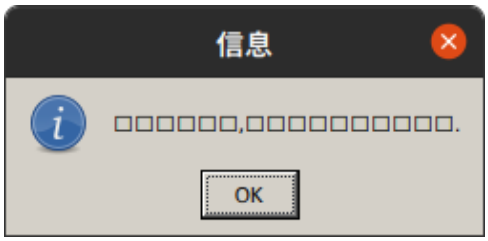

Figure 6: SpiImageTools final dialog

Click OK then exit the program. Now in the directory where SpiImageTools.exe is located there should be three new files: boot0.bin, boot1.bin, and data.bin. data.bin is the final flashable image. Congrats, the build process is now complete!

3.2 Extracting on Windows

In order to make the extraction process easier, Windows can be used to run the SpiImage-Tools executable. Start by transferring update.img to the Windows OS. Now double click on SpiImageTools.exe and the program will open the following window.

Figure 7: The SpiImageTools program

Click the top button to open a file dialog box, and navigate to the location of the update.img file that was copied over. Once the file is opened, click the bottom button and the blue bar will begin to ping back and forth for a little while. It will be complete when the following message appears.

Figure 8: SpiImageTools final dialog

Click OK then exit the program. In the directory where the executable is located there should be three new files: boot0.bin, boot1.bin, and data.bin. data.bin is the final flashable image. Congrats, the build process is now complete!

4 Flash

The following section describes how to flash the data.bin image onto the Tinkerboard for actual use, it is almost the same as the process described in the SMES Install Instructions Document. See that document for further Android setup after flashing.

4.1 Install Etcher

In order to flash the Android operating system onto the TinkerBoard's internal memory or SD Card, having flashing software is very useful. Other methods may be used, but Balena Etcher is proven to work in a convenient manner. To install it, go to the following link:

https://www.balena.io/etcher/

Download the application or installer and run it. Once Etcher is installed and running, the following window will appear.

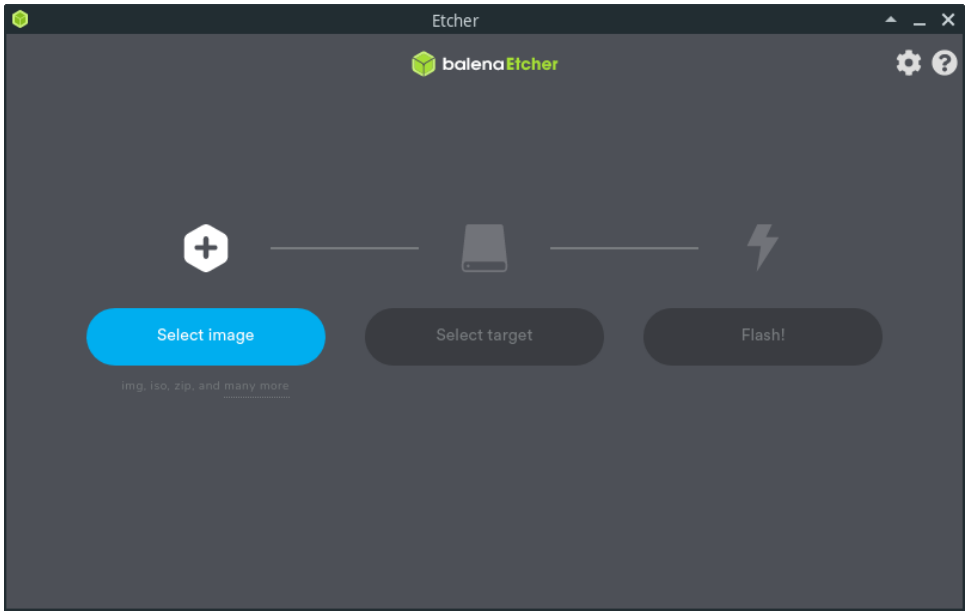

Figure 9: Etcher Home Screen

4.2 Flash OS

On the Etcher window, click the **Select Image** button and choose the data.bin file. Etcher may have a **Missing Partition Table** warning pop up. This is because the Android OS has a different partition scheme from most operating systems. Just click **Continue** to move on.

If using the TinkerBoard S, then follow the instructions in this paragraph to flash to eMMC. For the original TinkerBoard, skip to the next paragraph. Plug the tinkerboard in to the host system via a micro USB cable with working data lines. Usually, the device comes up automatically on the **Select Target** section after a few seconds of being plugged in. If not, click the button to try and find it as a connected USB device. If this does not work then the micro USB cable is probably bad. The section should now say something like "Linux Tin...Board_UMS" or "16 GB eMMC." Finally, click the **Flash** button to flash Android onto the TinkerBoard's internal flash memory. This may take a few minutes.

If using the original model Tinkerboard, then the OS must be flashed to a Micro SD card instead. To do this, Plug in a card via an adapter or SD port to the computer. Upon doing so, Etcher will often bring it up in the **Select Target** section automatically. If not, click the **Select Target** button and choose the Micro SD card via the file explorer. Then click the **Flash** button to flash Android to the card. Afterward, insert the Micro SD card into the slot on the TinkerBoard.

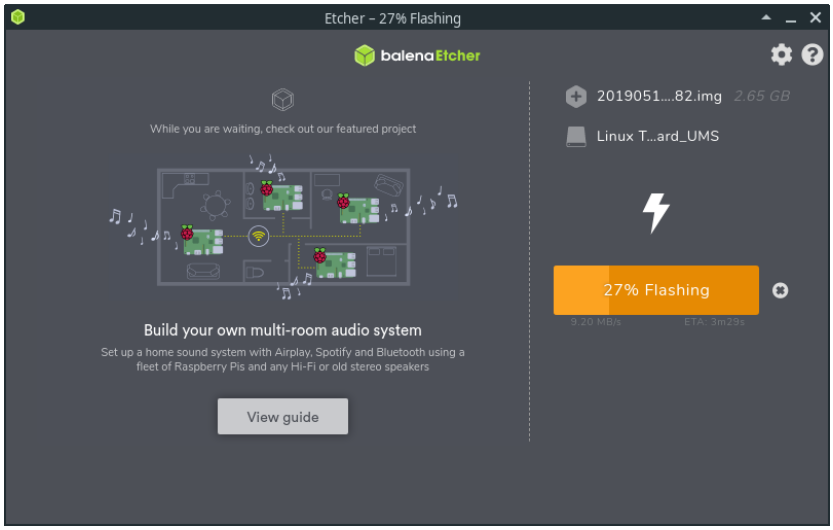

Figure 10: Etcher Flashing Screen

Once the flash is complete, the TinkerBoard should be ready to be used as an Android device. Plug in a monitor or TV to its HDMI port. Plug in a keyboard and mouse. Then plug in a 5V-3A microUSB power cable. The first startup may take a little while to boot.

5 Editing Source

There are a few important parts of the Android source tree to focus on for SMES. This includes Trusty TEE, GPIO and GPIO protocol control, and access to root. This section describes were to find these in the source tree and what files to edit for certain functionality.

5.1 Trusty TEE

The files responsible for the high level algorithms trusted execution environment can be found in the **kernel/security/optee_linuxdriver/core** folder within the Android source folder. While the ARM TrustZone drivers are found in **kernel/security/optee_linuxdriver/armtz**.

5.2 GPIO

5.3 Root Access