Patchrom Developer Guide

1. Introduction

Patchrom is a open-source project initiated and maintained by MIUI team. It mainly utilizes the assembler/disassembler technology to craft a MIUI ROM for an existing ROM. Although this project is for MIUI ROM, but the technology and tools is general for crafting any android ROM.

1.1 Setup Develop Environment

1.1.1 OS

We recommend to use Ubuntu(10.10 version or above is fine). Currently we have no plan to run patchrom project on Windows and Mac OS X.

1.1.2 Android SDK

Refer to http://developer.android.com/sdk/installing.html for how to install Android SDK.

After ANDROID SDK installation is complete, modify the PATH environment variable to provide access to SDK's tools/ and platform-tools.
Edit your ~/.bash_profile or ~/.bashrc file. Look for a line that sets the PATH environment variable and add the full path to the tools/ and platform-tools/ directories to it. If you don't see a line setting the path, you can add one: <sdk> is your android SDK installation directory.

export PATH=${PATH}:<sdk>/tools:<sdk>/platform-tools

1.1.3 Android Source

Patchrom doesn't require the android source code to build. But it will be better to understand and have the android source code for reference. See http://source.android.com/source/index.html for how to get and build the source code.

1.2 Download the source

```
$ mkdir patchrom
$ cd patchrom
$ repo init -u git://github.com/MiCode/patchrom.git -b ics; repo sync
```

1.3 Patchrom Walkthrough

Pachrom is used to craft MIUI ROM for android device. So for each specific device there is a standalone directory for this device. For example, currently we support Sumsung i9100, Huawei honor, Sony lt18i and HTC sensation.

And there are some directories which are common to all devices:

(1) android/: This directory contains all the stuff related to the baseline ics source code released by google.

(2) build/: This directory contains all the makfile for build.

(3) miui/: This directory contains all the APKs and 3 jars built from MIUI source code and also all the resources.

(4) tools/: This directory contains all the scripts, apps used to build the project. Especially, we used the apktool and smali/baksmali. Thanks to those guys for contributing them.

1.4 Build

Assume current directory is patchrom and we will build the ROM for Huawei Honor:

```
$ . build/envsetup.sh
$ cd honor
$ make fullota
```

After build completed, there will be a fullota.zip under out directory which is ready for flash into honor through recovery.

2. Preparation work

From now on, we want to clarify two concepts: recovery-ROM and ROM. When we talk about Recovery-ROM, we specifically mean a ROM which can be flashed into device in recovery mode. It is a zip file and conforms to the standard format required by recovery. When we talk about ROM, it is a general concept; for those file which can be flashed into your device to update the system is called a ROM. Also the flash method varies, e.g, Samsung's odin, Motorola's RSD, Sony uses fastboot etc. The patchrom project only generates recovery-ROM, since this kind of ROM is most general and flexible.

In order to successfully and smoothly craft miui recovery-ROM for your own device, it is important do some preparation work. In one word, the critical point is to find a suitable ROM for your device.

2.1 Familiar with your device

The first step is to familiar with your device, includes how to flash ROM, how to root and try some ROMs aimed at your device. The main work of this step is to surf some android forums which have rich information for your device. Here is a forum you definitely should look at: http://forum.xda-developers.com/

2.2 Choose suitable ROM

After getting acquainted with your device, be prepared to choose a suitable ROM for patchrom project. How to decide if a ROM is suitable, here are some guidelines:

(1) Android version must be in minor difference.

Currently, patchrom is based on the ics-4.0.3. The chosen ROM's version should be ics-4.0.1 to ics-4.0.3. The principle here is that the chosen ROM's android version's source code should be in minor change to the android version's source code used by patchrom.

(2) Stock ROM

The stock ROM which is released by device manufacturer is preferred. Since the stock ROM is most stable.

(3) Root

Generally speaking there are 2 kinds of root: root privilege and kernel root. The root privilege means that will be a Superuser app which will popup a dialog to ask if you approve the request when other apps require root privilege. The kernel root means you can directly manipulate this device in root privilege via adb. The kernel root is preferred.

(4) Recovery

Since patchrom generates a recovery-ROM, So be sure that your device can enter recovery mode. The CWM recovery and ext4 recovery are good choices.

3. Craft miui recovery-ROM

When a suitable ROM is found and has already been flashed into your device, we will start our crafting journey.

3.1 Generate base recovery-ROM

Firstly, we should create a standalone directory for the device. Let's use an imaginary device called xblade and assume current directory is patchrom.

$. build/envsetup.sh
$mkdir xblade
$ cd xblade

Secondly, we will run a tool to automatically generate a recovery-ROM from the running device. You may ask, why we bother to generate a recovery-ROM if we already find a suitable ROM. There are two reasons:

(1) Sometimes we can't find a stock recovery-ROM. But we requires a recovery-ROM as the basis.

(2) Most importantly, running this tool first is critical to automaticly generate miui recovery-ROM and incremental OTA which makes miui so special.

Now connect your device to PC, ensure the USB Debug mode is on.

$ adb reboot recovery (if this command doesn't work, manually reboot into recovery mode)

$ ../tools/ota_target_from_phone -r (-r means run this tool in recovery mode, we can also run this tool in normal mode, please refer to the source of this tool for more information)

This tool will generate a stockrom.zip and a metadata directory which will be used to generate our miui recovery-ROM.

3.2 makefile

We need a makefile to let make work. Here is a makefile to begin with:

```
local-zip-file          := stockrom.zip
local-out-zip-file      :=
local-miui-modifed-apps :=
local-modified-apps     :=
local-miui-removed-apps :=
local-phone-apps        :=
local-pre-zip           := local-zip-misc
include $(PORT_BUILD)/porting.mk
local-zip-misc:
```

We will explain some definitions in this file, for thorough understanding, you definitely should look at the whole build system.

- local-zip-file: the recovery-ROM generated in previous section, must be specified for each device.
- local-out-zip-file: the output recovery-ROM filename when you run "make zipfile"
- local-miui-modified-apps: all the miui apps are listed in the file patchrom/build/miuiapps.mk. This variable defines those miui apps which we modified.
- local-modified-apps: This variable defines those apps from local-zip-file which we modified.
- local-miui-removed-apps: Normally not all the miui apps is suitable for our device(e.g. some device will not want Phone.apk). This variable defines those miui apps which we don't want.

- local-phone-apps: This variable is used to remove some apps from local-zip-file. As you can see, we use blacklist for miui apps and whitelist for stock ROM apps. This variable defines those stock ROM apps which we want to keep.
- local-zip-misc: This target permits you to do some customization before we generate the final recovery-ROM.

3.3 workspace

The first make command we will run is "make workspace". This will prepare the workspace for you. The operation is simple, it will extract framework/android.policy/services.jar and framework-res.apk from stockrom.zip and run apktool to disassemble them.

3.4 firstpatch

The second make command we will run is "make firstpatch". Before diving into this command, we need to talk about smali.

We use apktool to disassemble the jar/apk which generates smali code. We will use framework.jar.out(which is the disassembled smali code from framework.jar) as an example to discuss this:

There are 3 smali code directories involved:
(1) old framework.jar.out
(2) new framework.jar.out
(3) target framework.jar.out
At first time, the old framework.jar.out will be the disassembled framework.jar which is compiled from google released source code, the new framework.jar.out will be the disassembled framework.jar which is compiled from miui's source code. The target framework.jar.out will be the disassembled framework.jar from target device which is already prepared in "make workspace".

In order to facilitate this process, we try the best to make minimum change to the google released source code. You can see these change by compare miui/src and android/src. But most often there will be some conflicts when applying the patch.

After running this command, there will be a temp directory. There will be 5 child directories:
(1) old_smali: the smali code from old framework.jar.out with .line removed
(2) new_smali: the smali code from new framework.jar.out with .line removed
(3) dst_smali_orig: the smali code from target framework.jar.out with .line removed
(4) dst_smali_patched: the smali code after applying the patch into target framework.jar.out with .line removed
(5) reject: the rejected patch. We should use this directory to resolve the conflicts.

You may raise a question: what if miui makes further change to the source code, how can we

track these changes.

Firstly, please record the newest commit number of android/framework.jar.out when you run "make firstpatch". Assume the commit number is abcdefg.

Then at some time, the android/framework.jar.out will be updated and assume the newest commit number is hijklmn. run "tools/git.patch abcdefg..hijklmn > patch.file".

Finally at your device directory, run "tools/git.apply < ../android/patch.file". Now you can resolve any conflict.

3.5 fullota

After resolved the conflicts, you can run "make fullota" to build the final recovery-ROM. This command will generate fullota.zip under out directory. Flash this file in recovery mode. And now your device may boot successfully and see the miui lockscreen. But most often things are not going so well, don't worry. "adb logcat" will be the great tool to help us. From now on, there will be a problem-solving loop. Solve the problem or exception we encountered.

3.6 suggestion

First of all, you should read the source code to understand the whole build system.

Secondly, you can refer to the devices we already finished, especially for the devices from same manufacturer.

Finally, please post your issue and we will try to answer it. It will be highly appreciate if you open source your device, you can use our server to release the ROM to users and enjoy the live MIUI through OTA update.