Breakpad steps

Introduction

- Crash dumps are a great source of information pointing to the reason of an unfortunate software crash
- RDK-B uses Google Breakpad to generate crash dumps

Google Breakpad

- A set of client-server components that implement a crash report system
- Consists of the below components
    - Client library – Attached to the target binary
    - Symbol table – Generated from the debug version using an associated tool call dumpsyms
    - Minidump file – Generated as part of a crash
    - Stackwalker – A tool to generate the stack trace using the symbol table and minidump

Advantages of breakpad

- Executable does not require to have debug information to generate dumps
- Uses proven and lighter 'minidump' format by windows

Breakpad function

Header file	#include "breakpad_wrapper.h"
Source file	breakpad_wrapper.cpp
Component	breakpad_wrapper
Callback function	breakpad_ExceptionHandler()
CFLAG(ccsp_common.inc)	INCLUDE_BREAKPAD
LDFLAG	-lbreakpadwrapper
Library	libbreakpadwrapper.la (libbreakpadWrapper.so)
Installing wrapper for breakpad (ccsp-common-library.bbappend)	/usr/include/breakpad
Packaging (packagegroup-rdk-ccsp-broadband.bb)	breakpad-wrapper \
Debug.ini	LOG.RDK.BREAKPAD=ALL FATAL ERROR WARNING INFO DEBUG

Steps to generate crash dump

- After successful compilation, copy the executable from VM to your board under /usr/bin/
- in your device, check whether /minidumps is available . If not available , do mkdir /minidumps/
- run the executable from /usr/bin
- dump file will be available under /minidumps

Steps to Decode crash dump

- In VM , clone Google breakpad utility

```
mkdir breakpad

cd breakpad

git clone https://chromium.googlesource.com/breakpad/breakpad breakpad_pc

cd breakpad_pc

git clone https://chromium.googlesource.com/linux-syscall-support src/third_party/lss

./configure

make
```

- Go to /breakpad
- Copy the crash dump from your board and place it under /breakpad
- Run dump_syms with stripped and not-stripped executables

```
breakpad_pc/src/tools/linux/dump_syms/dump_syms  <stripped> <not-stripped> > <sym file>
```

- head -n1 <sym file>
- mkdir -p symbols/<component>/<head>
- mv <sym file> symbols/<component>/<head>
- Run "minidump_stackwalk" tool on minidump file

```
breakpad_pc/src/processor/minidump_stackwalk <dump file>  symbols/ > dump
```

- Vi dump << has generated logs

Sample code:

- clone google breakpad
- cd ..
- copy stripped executable inside stripped folder , not-stripped executable to not-stripped folder , copy core dump from your board
  Stripped : build-raspberrypi-rdk-broadband/tmp/work/cortexa7t2hf-neon-vfpv4-rdk-linux-gnueabi/utopia/rdkb-20200207-r0/package/usr/bin/firewall
  Not-stripped : build-raspberrypi-rdk-broadband/tmp/work/cortexa7t2hf-neon-vfpv4-rdk-linux-gnueabi/utopia/rdkb-20200207-r0/package/usr/bin/.debug/firewall
  dump : copy from board
- breakpad_pc/src/tools/linux/dump_syms/dump_syms stripped/CcspTr069PaSsp not-stripped/ > CcspTr069PaSsp.sym => <component>.sym
  (sym file name should be same as your executable)
- head -n1 CcspTr069PaSsp.sym

```
MODULE Linux arm FB145C1D7658C035E5C92DCC55DE4C180 CcspTr069PaSsp
```

- mkdir -p ./symbols/CcspTr069PaSsp/FB145C1D7658C035E5C92DCC55DE4C180
- mv CcspTr069PaSsp.sym ./symbols/CcspTr069PaSsp/FB145C1D7658C035E5C92DCC55DE4C180
- breakpad_pc/src/processor/minidump_stackwalk <core.dump> ./symbols/ > dump.out