Chaining HALs

ABS 2015

PRELIMINARY

HY Research LLC
http://www.hy-research.com/
Mar 15, 2015

Agenda

* Introduction - Why chain a HAL?
* Android HAL basics
* Overview of chaining a HAL
* HAL loader
* Exampling: sensor HAL chaining
* Conclusion
* Questions

Why Chain a HAL?

* HAL: **H**ardware **A**bstraction **L**ayer
* Binary blob to talk to Hardware
* Bug work around a binary HAL
* Adding features to a binary HAL
* Modified Hardware
* Prototyping
* Reuse of code
* Obsolete code

Android HAL basics

* Connects HW to Android
  * GPS, Sensors, Graphics, Sound
  * Radio ()
* HALs offer a place to provide virtual HW
* HW specific nature may lead to binary HALs.
* HALs are shared ELF objects
  Uses a well known symbol, HMI

HY Research LLC
http://www.hy-research.com/
Mar 15, 2015

How does it work?

* Subsystem requests a HAL via hw_module_load() (hardware/libhardware)

```
err = hw_get_module(SENSORS_HARDWARE_MODULE_ID,
                    (hw_module_t const**)&module);
```

1) Search /system/lib/hw and /vendor/lib/hw for a file named MODULE.tag.so. Where:
   MODULE is a string from the first paramter.
   tag is a string from system properties:
   - ro.hardware
   - ro.poduct.board
   - ro.board.platform
   - ro.arch

   or fall back of "default"

How does it work? (con't)

2) If the file exists, the search stops!
    - Even if it is not valid!
    - Other combinations not tried.
3) File is opened with dlopen()
4) The HMI symbol is located with dlsym()
    - HAL_MODULE_INFO_SYM_AS_STR
    - HAL_MODULE_INFO_SYM

How does it work? (con't)

5) Basic sanity check.
id member of HMI is checked.

* After hw_module_load(), HAL specific initialization.

* HAL specific information is often overlayed onto the HMI structure.
  - Sensor HAL adds a get sensors callback

HAL Chaining Overview

* Security folks - Man In the Middle Attack
* Implement 2 interfaces:
  1) Standard HAL API loadable by Android
  2) HAL loading interface like Android.
* ELF tap dancing
  1) Potentially identical names
  2) Track using pointers
* File names:
  1) Rename and/or move original HAL
  2) Place new HAL in old place.

HY Research LLC
http://www.hy-research.com/
Mar 15, 2015

Other chaining

* Non Android userland using glibc
  - LD_PRELOAD
  - LD_LIBRARY_PATH
  - Done via the dynamic linker in glibc.
  - Not supported by bionic

Example: Sensor HAL

* Nothing special. Just easy to demo.
* 3 parts
  - a normal HAL skeleton for sensors
  - a simplified HAL loader
  - sensor specific details.
* Goal: modify select data coming from the real sensor HAL.

Sensor HAL skeleton

* Fulfill Android requirements
* Define the HMI structure
  - Provide an open callback that initializes methods for sensors.
  - Provide a get_sensor_list() callback returns a list of sensors.

Sensor HAL skeleton

```
static struct hw_module_methods_t chain_sensors_module_methods = {
        open: chain_open_sensors
};

struct sensors_module_t HAL_MODULE_INFO_SYM = {
        common: {
                tag: HARDWARE_MODULE_TAG,
                version_major: 1,
                version_minor: 0,
                id: SENSORS_HARDWARE_MODULE_ID,
                name: "Sensor module",
                author: "HY Research LLC",
                methods: &chain_sensors_module_methods,
        },
        get_sensors_list: chain_sensors__get_sensors_list,
};
```

Sensor HAL loader

* Simplified version of hw_module_loader()
* Directly dlopen() the original HAL. No searching.
* Use dlsym() to find the pointer to the HMI structure and save it off.
  - Starting point to find entries into the original HAL.
* Gets invoked by the open method of our HAL.

```
static int chain_open_sensors(const struct hw_module_t* module, const char* id,
                        struct hw_device_t** device)
{
        [Declarations removed/error handlign removed.]
        ...
        const char *sym = HAL_MODULE_INFO_SYM_AS_STR;

        snprintf(old_sensorHAL_path, 2048, "%s/sensors.old.so", "/system/lib/hw");
        handle = dlopen(old_sensorHAL_path, RTLD_NOW);
        ...
        old_hmi = (struct sensors_module_t *)dlsym(handle, sym);
        ...
        if (strcmp(SENSORS_HARDWARE_MODULE_ID, old_hmi->common.id) != 0) {
                dlclose(handle); return -EINVAL;
        }
        ChainHALinfo.old_handle = handle;
        ChainHALinfo.get_sensors_list = old_hmi->get_sensors_list;
        old_status = (old_hmi->common.methods->open)(&(old_hmi->common), id, device);
        old_device = (struct sensors_poll_device_t *)*device;
        ChainHALinfo.old_poll = old_device->poll;
        old_device->poll = chain_poll__poll;
        return old_status;
}
```

Modifying the data

* Replace the sensor HAL specific poll method with our own and save old poll method.
* New poll method:
  - Calls old poll method to acquire the data
  - Inspects the data for things to modify.
  - Modify data found.

```
static int chain_poll__poll(struct sensors_poll_device_t *dev,
        sensors_event_t* data, int count)
{
        int old_ret;
        int i;

        /* Acquire data */
        old_ret = ChainHALinfo.old_poll(dev, data, count);

        /* Modify data if needed */
        if (old_ret > 0) {
                /* There is data! */
                for (i = 0; i < old_ret; i++) {
                        if (data[i].type == SENSOR_TYPE_ACCELEROMETER) {
                                data[i].data[0] = -data[i].data[0];
                                data[i].data[1] = -data[i].data[1];
                                data[i].data[2] = -data[i].data[2];
                        }
                }
        }

        return old_ret;
}
```

Summary

* HAL chaining can work around limitations of binary blobs.
* HALs are ELF objects with a well known symbol, HMI
* To chain a HAL, 2 things needs to happen:
  1. The HAL interface for Android needs to be implemented.
  2. A HAL loader needs to be written.

Questions?

HY Research LLC
http://www.hy-research.com/
Feb 11, 2013