

PhD Diary

Wei Minn
Singapore Management University

Version 1.0.5
Last compiled: June 10, 2024

Contents

I 2023	1
1 November	3
1.1 November 14, 2023	3
1.1.1 AOSP	3
1.1.2 C++ Primer	6
1.2 November 15, 2023	8
1.2.1 C++ Primer	8
1.2.2 AOSP	11
1.3 November 17, 2023	13
1.3.1 AOSP Hardware Abstraction Layer	13
1.4 November 18, 2023	15
1.4.1 Configuring AOSP for Acme Device on Ubuntu 23	15
1.5 November 19, 2023	16
1.5.1 Building AOSP HAL	16
1.6 November 20, 2023	17
1.6.1 const Qualifier	17
1.6.2 Rudimentary AOSP HAL Application	20
1.7 November 21, 2023	21
1.7.1 Fixing SELinux Policy to Start System Service	21
1.7.2 Memory Dump	21
1.8 November 22, 2023	22
1.8.1 C++ Data Structure	22
1.9 November 23, 2023	24
1.9.1 C++ Types and Data Structures	24
1.10 November 25, 2023	25
1.10.1 Android Memory Dump	25
1.11 November 27, 2023	26
1.11.1 Android Profile Guided Compilation	26
1.12 November 29, 2023	27
1.12.1 Android OAT Dump Parser	27

2 December	31
2.1 December 01, 2023	31
2.1.1 C++ Strings and Vectors	31
2.2 December 03, 2023	37
2.2.1 C++ Iterator, and Arrays	37
2.3 December 04, 2023	43
2.3.1 C++ Expressions I	43
2.4 December 05, 2023	44
2.4.1 C++ Expressions II	44
2.5 December 06, 2023	50
2.5.1 C++ Statements	50
2.6 December 10, 2023	52
2.6.1 Learn about SELinux	53
2.7 December 13, 2023	59
2.7.1 Android Testing	59
2.8 December 19, 2023	60
2.8.1 SELinux Syntax	60
2.9 December 22, 2023	62
2.9.1 SELinux Practical	62
2.10 December 24, 2023	62
2.10.1 Run Daemon	62
2.11 December 25, 2023	64
2.11.1 Run Daemon	64
II 2024	67
3 January	69
3.1 January 1, 2024	69
3.1.1 Connect Daemon to ART	69
3.2 January 2-3, 2024	73
3.2.1 Connect ART to Bionic	73
3.3 January 4-7, 2024	81
3.3.1 Load Native Shared Libraries to Daemon	81
3.4 January 11-15, 2024	85
3.4.1 Dronlomaly Tool Demo Paper	85
3.5 January 16-20, 2024	89
3.5.1 Explore Migration of Zicheng's Works to AOSP14	89
3.6 January 21-31, 2024	90
3.6.1 Understand Comprehensive Android Invocation Flow	90

4	Feubuary	95
4.1	Feubuary 11-12, 2024	95
4.1.1	Understand Papers on Android Framework Level	95
4.2	Feubuary 13, 2024	98
4.2.1	Conduct Experiments on AOSP 14	98
4.3	Feubuary 14-15, 2024	100
4.3.1	Figure out NatiDroid for AOSP14	100
5	March	103
5.1	March 05-13, 2024	103
5.1.1	Migrate Zicheng's code	103
5.2	March 14-30, 2024	110
5.2.1	Migrate Zicheng's code	110
5.3	March 31, 2024	112
5.3.1	Troubleshoot DynDroid on AOSP 14	112
6	April	119
6.1	April 01, 2024	119
6.1.1	Complete Lifecycle Hook Execution Flow	119
6.2	April 02-07, 2024	121
6.2.1	Complete ART Execution Flow	121
6.3	April 08, 2024	121
6.3.1	Meeting with iSprint	121
6.4	April 09-15, 2024	123
6.4.1	Recompile AOSP 13	123
6.4.2	Which components are involved in ART?	123
6.4.3	Components of Debloater Flow in AOSP 13	123
6.4.4	Test Customized Code without the External Headers	129
6.4.5	ART Loading Process of OAT Files	129
6.5	April 16-29, 2024	132
6.5.1	ART Loading Process of Classes and Methods	132
6.5.2	ART Execution Process of Classes and Methods	134
7	May	137
7.1	May 05-06, 2024	137
7.1.1	Run Zicheng's code without the weird text files	137
7.1.2	Implement Zicheng Code inside AOSP13 Emulator	150
7.2	May 07-14, 2024	150
7.2.1	Verify AOSP13 Debloating via ART Only	151
7.2.2	Which components are involved in Framework for Debloating? .	151
7.3	Isolate functionalities of ART Procedures	151
7.4	May 15, 2024	157

7.4.1	Verify debloater in several android apps	157
7.5	May 19, 2024	162
7.5.1	What makes up FjordPhantom exactly?	162
7.6	May 22-27, 2024	164
7.6.1	Repackaging	164
7.6.2	Virtualization and Containers	165
8	June	169
8.1	June 10, 2024	169
8.1.1	Decompile APKs	169
8.1.2	Repackaging Attack	169
8.1.3	App Virtualization Attack	171

Part I

2023

Chapter 1

November

1.1 November 14, 2023

1.1.1 AOSP

Zygote

Zygote initializes by pre-loading the entire Android framework. Unlike desktop Java, it does not load the libraries lazily; it loads all of them as part of system start up. After completely initializing, it enters a tight loop, waiting for connections to a socket. When the system needs to create a new application, it connects to the Zygote socket and sends a small packet describing the application to be started. Zygote clones itself, creating a new kernel-level process.

Memory is organized into uniformly sized **pages**. When the application refers to memory at a particular address, the device hardware reinterprets the address as an index into a **page table**. Newly cloned Zygote processes for newly started applications are simply clone of Zygote's page table, pointing to the exact same pages of physical memory. Only the pages the new application uses for its own purposes are not shared:

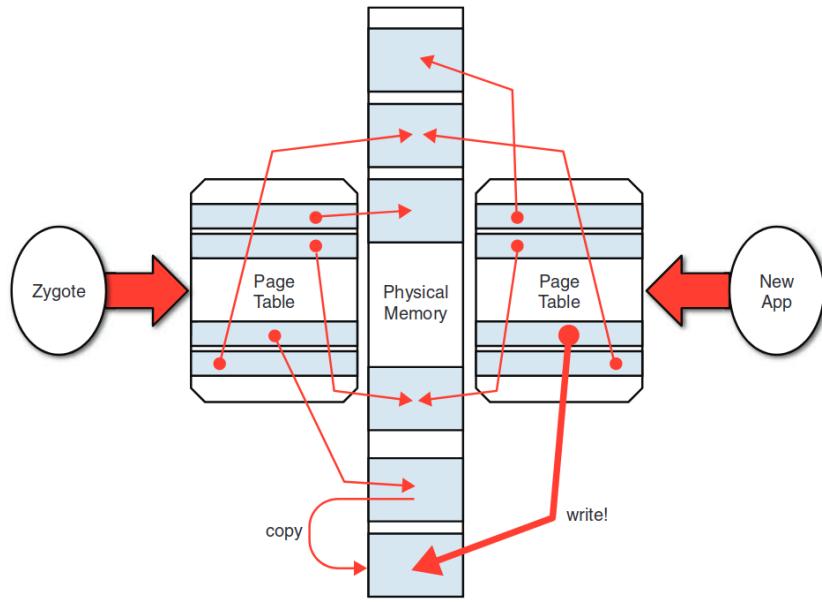


Figure 1.1: Zygote Copy-on Write

Zygote Initialization

Zygote is started by `init. ro.zygote` system variable set at platform build time decides which of four types of Zygotes are started and which one is "primary". Both the `init` and Zygote scripts are stored inside `$AOSP/system/core/rootdir`. In the following `init.zygote64_32.rc`, 2 Zygote processes, primary and secondary, are started at 2 different sockets:

```

1  service zygote /system/bin/app_process64 -Xzygote \
2    /system/bin --zygote --start-system-server --socket-name=zygote
3    class main
4    priority -20
5    user root
6    group root readproc reserved_disk
7    socket zygote stream 660 root system
8    socket usap_pool_primary stream 660 root system
9    onrestart exec_background - system system -- /system/bin/vdc volume abort_fuse
10   onrestart write /sys/power/state on
11   onrestart restart audioserver
12   onrestart restart camerасerver
13   onrestart restart media
14   onrestart restart media.tuner
15   onrestart restart netd
16   onrestart restart wificond
17   task_profiles ProcessCapacityHigh MaxPerformance
18   critical window=${zygote.critical_window.minute:-off} target=zygote-fatal
19
20  service zygote_secondary /system/bin/app_process32 -Xzygote \
21    /system/bin --zygote --socket-name=zygote_secondary --enable-lazy-preload
22    class main

```

```

23 priority -20
24 user root
25 group root readproc reserved_disk
26 socket zygote_secondary stream 660 root system
27 socket usap_pool_secondary stream 660 root system
28 onrestart restart zygote
29 task_profiles ProcessCapacityHigh MaxPerformance
30 disabled

```

The actual application that is started as user root at the very highest priority by init is /system/bin/app_process64. The script requests that init create a stream socket for the process and catalog it as /dev/socket/zygote_secondary which will be used by the system to start new Android applications.

Zygote is only started once during the system startup, by app_process64 and app_process32, and is simply cloned to start subsequent applications. Zygote initialization sequence is described below:

Method	Description	Source
init.rc	Imports the init.zygote64_32.rc that contains the script that starts Zygote service.	\$AOSP/system/core/rootdir
init.zygote64_32.rc	Runs app_process64 and app_process32 which will initialize the starting of Zygote service.	\$AOSP/system/core/rootdir
app_process	Creates AppRuntime, a subclass of AndroidRuntime, that does bookkeeping, naming the process, setting up parameter, and the name of the class to run when not running Zygote, and then calls AndroidRuntime.start() to invoke the runtime.	\$AOSP/frameworks/base/cmds/app_process
AppRuntime::start	Invokes startVM which invokes JNI_CreateJavaVM.	AOSP/frameworks/base/core/jni/AndroidRuntime.cpp
JNI_CreateJavaVM	Calls Runtime::Create.	\$AOSP/art/runtime/jni/java_vm_ext.cc
Runtime::Create	Initializes the ART runtime, loading the system OAT files and the libraries they contain.	\$AOSP/art/runtime/runtime.cc

Table 1.1: Zygote Initialization Sequence

The argument that app_process passed to start is com.android.internal.os.Zygote.Init, the source for which is in \$AOSP/frameworks/base/core/java/com/android/inter-

nal/os/ZygoteInit.java. `app_process` is the launcher for all Java programs (not apps!) in the Android system, and Zygote is one example of the programs (system service) to be launched.

Zygote System Service

Zygote has 3 major tasks, on startup:

1. Register the socket to which the system will connect to start new application. Handled by `registerServerSocket` method which creates socket using the named passed as parameter for `init` script.
2. Preload Android resources (classes, libraries, resources and even WebViews) with a call to `preload` method. After `preload` is finished, Zygote is fully initialized and ready to clone to new applications very quickly.
3. Start Android System Server. Thus, `SystemServer` is the first application to be cloned by Zygote.

After it has completed these three tasks, it enters a loop, waiting for connections to the socket.

1.1.2 C++ Primer

Primitive Built-in Types

Includes **arithmetic types** and a special type named **void** which has no associated values and can be used in only a few circumstances, most commonly as the return type for functions that do not return a value.

The arithmetic types are divided into two categories: **integral types** (which include character and boolean types) and floating-point types.

Type	Meaning	Minimum Size
<code>bool</code>	boolean (true or false)	NA
<code>char</code>	character	8 bits
<code>w_char_t</code>	wide character	16 bits
<code>char16_t</code>	Unicode character	16 bits
<code>char32_t</code>	Unicode character	32 bits
<code>short</code>	short integer	16 bits
<code>int</code>	integer	16 bits
<code>long</code>	long integer	32 bits
<code>long long</code>	long integer	64 bits
<code>float</code>	single-precision floating-point	6 significant digits

double	double-precision floating-point	10 significant digits
long double	extended-precision floating-point	10 significant digits

Table 1.2: Zygote Initialization Sequence

Except for `bool` and extended character types, the integral types may be `signed` (can represent negative or positive numbers) or `unsigned`. By default, `int`, `short`, `long`, `long long` are all signed. To declare `unsigned` type, prepend `unsigned` to the type. `char` is signed on some machine and `unsigned` on others, and `unsigned int` is abbreviated as `unsigned`.

Conversions happen automatically when we use an object of one type where an object of another type is expected.

```
1 unsigned u = 10;
2 int i = -42;
3 std::cout << i + i << std::endl; // prints -84
4 std::cout << u + i << std::endl; // if 32-bit ints, prints 4294967264
```

In the above snippet, converting a negative number to `unsigned` will cause the value to "wrap around" because `unsigned` values can never be less than 0. Thus, extra care should be taken if we want to write loops with `unsigned` values and stopping conditions at negative values like the snippet below:

```
1 // WRONG: u can never be less than 0; the condition will always succeed
2 for (unsigned u = 10; u >= 0; --u)
3     std::cout << u << std::endl;
```

As such it is always advisable to not mix `signed` and `unsigned` types. By default, integer literals (42) are signed, while octal (024) and hexadecimal (0x14) may be signed or `unsigned`.

Escape sequences are used as if they were single characters:

```
1 std::cout << '\n'; // prints a newline
2 std::cout << "\tHi!\n"; // prints a tab followed by "Hi!" and a newline
```

Variables

Initialization is not assignment. Initialization happens when a variable is given a value when it is created. Assignment obliterates an object's current value and replaces that value with a new one.

Four different ways to initialize:

```
1 int units_sold = 0;
2 int units_sold = {0}; // list initialization; does not work for built-in types if
   // data loss is likely
3 int units_sold{0};
4 int units_sold(0);
```

Variables defined outside any function body are initialized to zero by default. Variables of built-in type defined inside a function are **uninitialized** and therefore undefined. Objects of class type that we do not explicitly initialize have a value that is defined by the class.

A **declaration** makes a name known to the program. A file that wants to use a name defined elsewhere includes a declaration for that name. A **definition** creates the associated entity. A definition involves declaration, allocates storage and may provide the variable with an initial value.

```

1 extern int i;      // declares but not define j
2 int j;           // declares and defines j
3 int k = 12;       // declares, defines and initializes j
4 extern double pi = 3.14;    // definition

```

Variables must be defined only once but can be declared several times. To use a variable in more than one file requires declarations that are separate from the variable's definition. To use the same variable in multiple files, we must define that in one - and only one - file. Other files that use that variable must declare - but not define - that variable.

1.2 November 15, 2023

1.2.1 C++ Primer

Scopes of Names

Most scopes in C++ are delimited by curly braces.

```

1 #include <iostream>
2 int main() {
3     int sum = 0;
4     // sum values from 1 through 10 inclusive
5     for (int val = 1; val <= 10; ++val)
6         sum += val; // equivalent to sum=sum+val
7     std::cout << "Sum of 1 to 10 inclusive is "
8         << sum << std::endl;
9
10 }

```

In above program, `main` - like most names defined outside a function - has **global scope** and thus, is accessible throughout the program. `sum` has **block scope** and is accessible from its point of declaration throughout the rest of the `main` function. `val` is defined in the scope of the `for` statement and can be used in that statement but not elsewhere in `main`.

Names declared in the outer scope can also be redefined in an inner scope although it is always a bad idea:

```

1 #include <iostream>
2 // Program for illustration purposes only: It is bad style for a function

```

```

3 // to use a global variable and also define a local variable with the same name
4 int reused = 42; // reused has global scope
5 int main() {
6     int unique = 0; // unique has block scope
7
8     // output #1: uses global reused; prints 42 0
9     std::cout << reused << "u" << unique << std::endl;
10
11    int reused = 0; // new, local object named reused hides global reused
12    // output #2: uses local reused; prints 0 0
13    std::cout << reused << "u" << unique << std::endl;
14
15    // output #3: explicitly requests the global reused; prints 42 0
16    std::cout << ::reused << "u" << unique << std::endl;
17
18    return 0;
19 }
```

When the scope operator (`:: operator`) has an empty LHS, it is a request to fetch the name on the RHS from the global scope.

References

A **reference** defines an alternative name for an object. A reference type can be defined by writing a declarator of the form `&d` where `d` is the name being declared:

```

1 int ival = 1024;
2 int &refVal = ival; // refVal refers to (is another name for) ival
3 int &refVal2; // error: a reference must be initialized
```

When we define a reference, instead of copying the initializer's value, we bind the reference to its initializer. Once initialized, a reference remains bound to its initial object. There is no way to rebind a reference to refer to a different object. Because there is no way to rebind a reference, references must be initialized.

A reference is not an object. Instead, a reference is just another name for an already existing object. Thus, *all* operation on that reference are actually operations on the object to which the reference is bound:

```

1 refVal = 2; // assigns 2 to the object to which refVal refers, i.e., to ival
2 int ii = refVal; // same as ii=ival
```

Because references are not objects, we may not define a reference to a reference. We can define references in a single definition with each identifier that is a reference being preceded by the `&` symbol:

```

1 int i = 1024, i2 = 2048; // i and i2 are both ints
2 int &r = i, r2 = i2; // r is a reference bound to i; r2 is an int
3 int i3 = 1024, &ri = i3; // i3 is an int; ri is a reference bound to i3
4 int &r3 = i3, &r4 = i2; // both r3 and r4 are references
```

Pointers

Like references, pointers are used for indirect access to other objects. Unlike a reference, a pointer is an object in its own right. Pointers can be assigned and copied; a single pointer can point to several different objects over its lifetime. Unlike a reference, a pointer need not be initialized at the time it is defined. Like other built-in types, pointers defined at block scope have undefined value if they are not initialized.

We define a pointer type by writing a declarator of the form `*d`, where `d` is the name being defined. The `*` must be repeated for each pointer variable:

```
1 int *ip1, *ip2; // both ip1 and ip2 are pointers to int
2 double dp, *dp2; // dp2 is a pointer to double; dp is a double
```

A pointer holds the address of another object. We get the address of an object by using the address-of operator (`& operator`):

```
1 int ival = 42;
2 int *p = &ival; // p holds the address of ival; p is a pointer to ival
3
4 double dval;
5 double *pd = &dval; // ok: initializer is the address of a double
6 double *pd2 = pd; // ok: initializer is a pointer to double
7 int *pi = pd; // error: types of pi and pd differ
8 pi = &dval; // error: assigning the address of a double to a pointer to int
```

We can use the dereference operator (`* operator`) to access that object:

```
1 int ival = 42;
2 int *p = &ival; // p holds the address of ival; p is a pointer to ival
3 cout << *p; // * yields the object to which p points; prints 42
4
5 *p = 0; // * yields the object; we assign a new value to ival through p
6 cout << *p; // prints 0
```

When we assign to `*p`, we are assigning to the object to which `p` points. We may dereference only a valid pointer that points to an object.

`void*` is a special pointer type that can hold the address of any object. Its useful for when the type of the object at that address is unknown:

```
1 double obj = 3.14, *pd = &obj; // ok: void* can hold the address value of any data
   pointer type
2 void *pv = &obj; // obj can be an object of any type
3 pv = pd; // pv can hold a pointer to any type
```

The modifiers `*` and `&` do not apply to all variables defined in a single statement:

```
1 int* p1, p2; // p1 is a pointer to int; p2 is an int
2 int *p1, *p2; // both p1 and p2 are pointers to int
```

As pointers are objects in memory, they also have addresses of their own. Therefore, we can store the address of a pointer in another pointer:

```
1 int ival = 1024;
2 int *pi = &ival; // pi points to an int
3 int **ppi = &pi; // ppi points to a pointer to an int
```

We indicate each pointer level by its own `*`. Dereferencing a pointer to a pointer yields the pointer. So in this case, you must dereference twice to access the underlying object.

1.2.2 AOSP

Starting Android System Server and Other Apps using Zygote

During its initialization, Zygote will check for `start-system-server` flag, and if set, will bring up `SystemServer` in the following sequence:

Method	Description	Source
<code>ZygoteInit.forkSystemServer</code>	Runs after the Zygote process has been initialized. It is hardcoded with System Server classpath ¹ as one of the arguments to call <code>Zygote.forkSystemServer</code> to spawn <code>SystemServer</code> process.	AOSP/framework/base/core/java/com/android/internal/os/ZygoteInit.java
<code>Zygote.forkSystemServer</code>	Zygote class wraps native methods that communicate with Android Runtime, one of whom is <code>com_android_internal_os_nativeForkSystemServer</code> .	AOSP/framework/base/core/java/com/android/internal/os/Zygote.java
<code>com_android_internal_os_nativeForkSystemServer</code>	Calls <code>zygote::ForkCommon</code> and <code>SpecializeCommon</code> which does the actual forking.	AOSP/frameworks/base/core/jni/com_android_internal_os_Zygote.cpp
<code>SpecializeCommon</code>	Looks at the flags and Process ID for setting up sandboxing, configuring the correct SE Linux context, and process capabilities. Afterwards, it will call Zygote methods for post-fork procedures.	AOSP/frameworks/base/core/jni/com_android_internal_os_Zygote.cpp
<code>Zygote.callPostForkSystemServerHooks</code>	Calls <code>ZygoteHooks.java</code> at the end of specialization procedures. Only applicable for <code>SystemServer</code> .	AOSP/framework/base/core/java/com/android/internal/os/Zygote.java

¹`com.android.server.SystemServer`, the source for which is stored in `AOSP/frameworks/base/services/java/com/android/server/SystemServer.java`.

Method	Description	Source
Zygote. callPostForkChildHooks	Calls ZygoteHooks.java at the end of specialization procedures. Applicable to all applications and services including SystemServer	AOSP/framework/base/ core/java/com/android/ internal/os/Zygote. java
ZygoteHooks. postForkSystemServer	Wrappers for ZygoteHooks inside the Android Runtime. They call and ZygoteHooks. their respective native code inside postForkSystemServer the ART via Java Native Interface.	AOSP/libcore/dalvik/ src/main/java/dalvik/ system/ZygoteHooks. java
ZygoteHooks_. nativePostForkSystemServer	Loads the specialized class libraries to start the the System Server.	AOSP/art/runtime/ native/dalvik_system_ ZygoteHooks.cc
ZygoteHooks_. nativePostForkChild	Loads the specialized class libraries to start the service/application.	AOSP/art/runtime/ native/dalvik_system_ ZygoteHooks.cc
handleSystemServerPro	These control returns to ZygoteInit, and finish remaining work for the newly forked system server process, and calls ZygoteInit.zygoteInit.	AOSP/framework/ base/core/java/com/ android/internal/os/ ZygoteInit.java
ZygoteInit. zygoteInit	The main function called when started through the zygote process, which calls RuntimeInit.applicationInit	AOSP/framework/ base/core/java/com/ android/internal/os/ ZygoteInit.java
RuntimeInit. applicationInit	Calls the public static void main method of the application	AOSP/framework/ base/core/java/com/ android/internal/os/ RuntimeInit.java
ZygoteServer. runSelectLoop	After forking has finished, the control enters ZygoteServer which starts an endless loop that handles incoming connections with ZygoteConnection.processCommand.	AOSP/framework/ base/core/java/com/ android/internal/os/ ZygoteServer.java
ZygoteConnection. processCommand	Calls Zygote.forkAndSpecialize which is a version of ZygoteInit.forkSystemServer for the masses.	AOSP/framework/ base/core/java/com/ android/internal/os/ ZygoteConnection.java
Zygote. forkAndSpecialize	A version of Zygote.forkSystemServer for the masses.	AOSP/frameworks/base/ core/jni/com_android_ internal_os_Zygote.cpp

Method	Description	Source
com_android_internal_os_native-ForkAndSpecialize	Calls zygote::ForkCommon and SpecializeCommon which does the actual forking, and returns to ZygoteInit and immediately enters ZygoteServer.	AOSP/frameworks/base/core/jni/com_android_internal_os_Zygote.cpp

Table 1.3: System Server and Applications Initialization Sequence

1.3 November 17, 2023

1.3.1 AOSP Hardware Abstraction Layer

The interface to the hardware is a device drivers which are usually device specific and sometimes proprietary. A single set of C header files describes the functionality that a HAL provides to the Android system. HAL Code for a particular device is the implementation of the API defined by those header files, so that no code above the HAL needs to be changed to port Android to use the new device.

HAL Code Structure

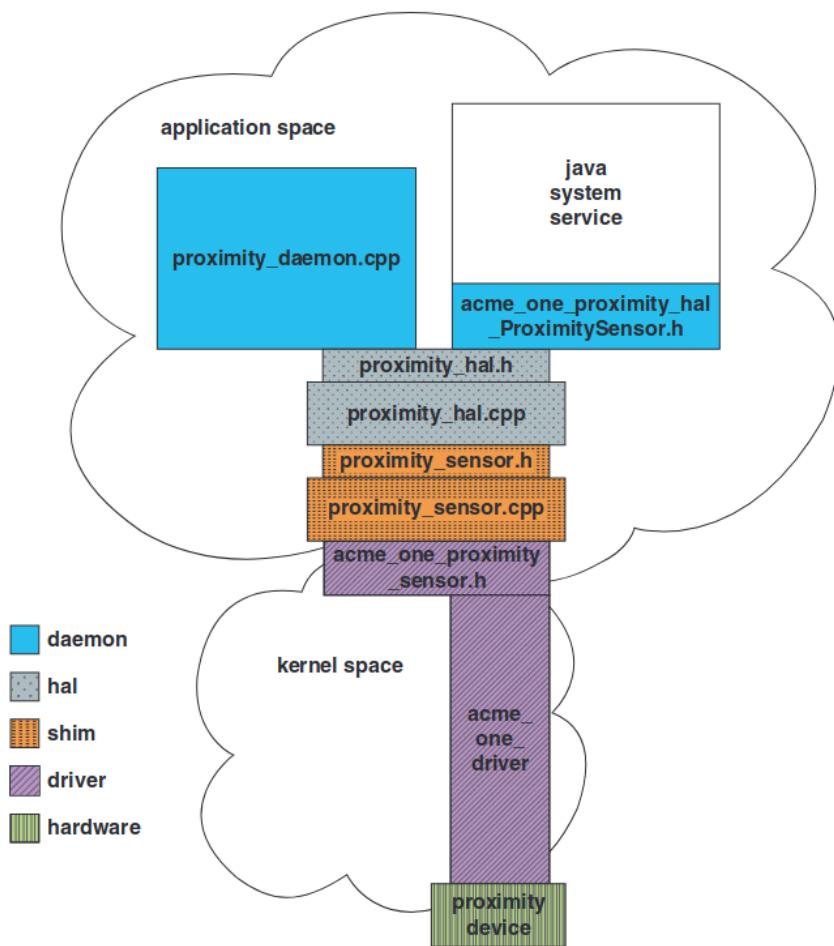


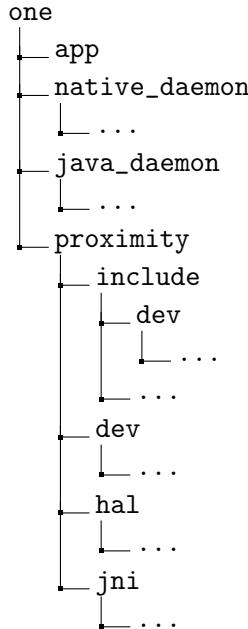
Figure 1.2: HAL Layer Structure

The code consists of four functional components as shown in Figure 1.2:

1. **HAL code (dotted boxes):** Abstraction that separates the capabilities of hardware from its specific implementations. The .h file defines the HAL interface, and the implementation (.cpp file) specializes the Android HAL API for the target hardware.
2. **Shim code (dashed boxes):** Glue code that connects the HAL to a specific device hardware/driver. This code adapts the Android HAL API to the device driver for the hardware.
3. **Daemon (blue):** Stand-alone application that interacts with the hardware through the HAL.

4. **Java System Service (white):** System Service that Android applications will use to access the custom hardware.

The source for those components are structured like in the directory tree below:



where one is the device folder of the AOSP project. All the code implementing the HAL for the proximity sensors goes into a new subdirectory `proximity`.²

1.4 November 18, 2023

1.4.1 Configuring AOSP for Acme Device on Ubuntu 23

Repo Manifest

Top-level subdirectory named `.repo` contains the `manifests` repository inside `manifests` subdirectory. The `manifests` repo contain one or more manifest files named as the argument of the `-m` command line option. `.repo/manifest` file controls the structure of the rest of the repository, and includes `.repo/manifests/default.xml`³ which is a list of git repositories. `repo` program parses `manifest.xml`, and thus `default.xml`, and clone each repository into a location specified inside `default.xml`.

²To be a "real" HAL, the interface `proximity/include/proximity_hal.h` would have to be promoted from its current directory specifically for the One device, up into the Android source tree to a location that would make it visible to other code that needed to use it. Here, it is only shared by Acme devices, so it is put under the subdirectory of the Acme device directory. If it's visible across device from multiple vendors, it might be promoted into the `device` directory itself.

³<https://gerrit.googlesource.com/git-repo/+/master/docs/manifest-format.md>

Each project element in the XML identifies a git repository by its name, relative to some base URL, its remote; and where that repository should be placed in the local workspace, its path. If the full URL for the repository is not specified, repo will use the default remote specified in the default element near the top of the manifest:

```
1 <default revision="refs/tags/android-13.0.0_r11"
2   remote="aosp"
3   sync-j="4" />
```

where the remote, aosp, is defined likewise in the top of of the default.xml:

```
1 <remote name="aosp"
2   fetch=".."
3   review="https://android-review.googlesource.com/" />
```

Instead of including a URL as its attribute, it includes the fetch attribute which indicates the URL for this remote should be derived from the URL used to initialize the workspace (the argument to the -u option).

```
1 git ls-remote -h https://android.googlesource.com/platform/manifest.git
2 repo init -u https://android.googlesource.com/platform/manifest -b android-10.0.0
3   _r33
4 git config --global user.email "mg.weiminn@gmail.com"
5 git config --global user.name "weiminn"
6 repo init -u https://android.googlesource.com/platform/manifest -b android-10.0.0
7   _r33
8 repo sync -j31
9 source build/envsetup.sh
10 lunch sdk_phone_x86_64-userdebug
11 make -j31
```

Ubuntu 23 does not have the repository for libncurses5 because they already have libncurses6, so you have to add old focal repository to your /etc/apt/sources.list:

```
1 deb http://security.ubuntu.com/ubuntu focal-security main universe
```

After including the old repo, update the repository index and install libncurses5:

```
1 sudo apt update
2 sudo apt install libncurses5
```

1.5 November 19, 2023

1.5.1 Building AOSP HAL

Implementing the HAL

Device driver and its API are usually provided by a third-party hardware provider. Shim code include .h files for one or more device drivers.

New devices can communicate with the processor via USB which is popular. Even without driver, the device can be accessed with generic USB commands. *libusb*⁴ is

⁴<https://libusb.info>

a portable, user-mode, and USB-version agnostic library for using USB devices, and supports Android. If the device has a driver, it is likely to be a specialization of the USB command.

```

1 #ifndef PROXIMITY_HAL_H
2 #define PROXIMITY_HAL_H
3
4 #include <hardware/hardware.h>
5
6 #define ACME_PROXIMITY_SENSOR_MODULE "libproximityhal"
7
8 typedef struct proximity_sensor_device proximity_sensor_device_t;
9
10 struct value_range {
11     int min; int range;
12 };
13
14 typedef struct proximity_params {
15     struct value_range precision;
16     struct value_range proximity;
17 }
18
19 proximity_params_t;
20
21 struct proximity_sensor_device { hw_device_t common;
22     int fd;
23     proximity_params_t params;
24     int (*poll_sensor)(proximity_sensor_device_t *dev, int precision);
25 }; #endif // PROXIMITY_HAL_H

```

1.6 November 20, 2023

1.6.1 const Qualifier

When we have variables whose value we know cannot be changed, and we want to prevent code from inadvertently giving a new value to the variable, we define the variable's type as `const`:

```

1 const int bufSize = 512; // input buffer size
2 bufSize = 512; // error: attempt to write to const object

```

Because we can't change the value of a `const` object after we create it, it must be initialized:

```

1 const int i = get_size(); // ok: initialized at run time
2 const int j = 42; // ok: initialized at compile time
3 const int k; // error: k is uninitialized const

```

const is Local to the File

When a `const` object is initialized from a compile-time constant as in:

```
1 const int bufSize = 512;
```

the compiler will usually replace uses of the variable with its corresponding value during compilation. Thus, when we split a program to multiple files, every file that uses that `const` must have access to its initializer. In order to see the initializer, the variable must be defined in every file that wants to use the variable's value. To support this, we define the `const` in one file, and declare it in other files that use that object.

```

1 // file_1.cc defines and initializes a const that is accessible to other files
2 extern const int bufSize = fcn();
3
4 // file_1.h
5 extern const int bufSize; // same bufSize as defined in file_1.cc

```

Because `bufSize` is `const`, we must specify `extern` in order for `bufSize` to be used in other files. `extern` signifies that `bufSize` is not local to this file and that its definition will occur elsewhere.

References to `const`

Unlike an ordinary reference, a reference to `const` cannot be used to change the object to which the reference is bound:

```

1 const int ci = 1024;
2 const int &r1 = ci; // ok: both reference and underlying object are const
3 r1 = 42; // error: r1 is a reference to const int &r2 =
4 ci; // error: nonconst reference to a const object

```

We can bind a reference to a `const` to a nonconst object, literals, or a more general expression:

```

1 int i = 42;
2 const int &r1 = i; // we can bind a const int& to a plain int object
3 const int &r2 = 42; // ok: r1 is a reference to const
4 const int &r3 = r1 * 2; // ok: r3 is a reference to
5 const int &r4 = r * 2; // error: r4 is a plain, nonconst reference

```

It is important to realize that a reference to `const` restricts only what we can do through that reference. Binding a reference to `const` to an object says nothing about whether the underlying object itself is `const`.

Pointers and `const`

Like a reference to `const`, a pointer to `const` may not be used to change the object to which the pointer points. We may store the address of a `const` object only in a pointer to `const`, where we can modify the pointer itself (change the reference stored inside) but not the object (value) pointed to:

```

1 const double pi = 3.14; // pi is const; its value may not be changed
2 double *ptr = &pi; // error: ptr is a plain pointer
3 const double *cptr = &pi; // ok: cptra may point to a double that is const
4 *cptr = 42; // error: cannot assign to *cptr

```

Like a reference to `const`, a pointer to `const` says nothing about whether the object to which the pointer points is `const`, that there is no guarantee that an object pointed to by a pointer to `const` won't change.

We can have a pointer that is itself `const`, and the address it holds cannot be changed. We indicate that the pointer is `const` by putting the `const` by the `const`:

```

1 int errNumb = 0;
2 int *const curErr = &errNumb; // curErr will always point to errNumb
3 const double pi = 3.14159;
4 const double *const pip = &pi; // pip is a constpointer to a const object
5
6 *pip = 2.72; // error: pip is a pointer to const and cannot be changed as the
    pointer is const
7 // if the object to which curErr points (i.e., errNumb) is nonzero
8 if (*curErr) {
9     errorHandler();
10    *curErr = 0; // ok: reset the value of the object to which curErr is bound
        because errNumb is not const
11 }
```

Constant Expressions

A constant expression is an expression whose value cannot change and that can be evaluated at compile time. A literal is a constant expression. A `const` object that is initialized from a constant expression is also a constant expression:

```

1 const int max_files = 20; // max_files is a constant expression
2 const int limit = max_files + 1; // limit is a constant expression
3 int staff_size = 27; // staff_size is not a constant expression
4 const int sz = get_size(); // sz is not a constant expression
```

Although `staff_size` is initialized from a literal, it is not a constant expression because it is a plain `int`, not a `const int`. Even though `sz` is a `const`, the value of its initializer is not known until run time, and thus, `sz` is not a constant expression.

We can ask the compiler to verify that a variable is a constant expression by declaring the variable in a `constexpr` declaration. Variables declared as `constexpr` are implicitly `const` and must be initialized by constant expressions:

```

1 constexpr int mf = 20; // 20 is a constant expression
2 constexpr int limit = mf + 1; // mf+1 is a constant expression
3 constexpr int sz = size(); // ok only if size is a constexpr function
```

Because a constant expression is one that can be evaluated at compile time, only literal types (arithmetic, reference, and pointer) types can be defined as constant expressions. Custom classes, library IO and string types are not literal types. We can point (or bind) to an object that remains at a fixed address.

`constexpr` declaration applies to the pointer, not the type to which the pointer points:

```

1 const int *p = nullptr; // p is a pointer to a constint
2 constexpr int *q = nullptr; // q is a constpointer to int
```

`p` is a pointer to `const` (low-level), whereas `q` is a constant pointer (top-level).

1.6.2 Rudimentary AOSP HAL Application

Create Rudimentary Service source file, rudi.cpp, in AOSP/device/generic/goldfish/app/wei_daemon

```

1 #include <unistd.h>
2 #include <stdio.h>
3 #include <android/log.h>
4
5 #define DELAY_SECS 2
6 #define ALOG(msg) __android_log_write(ANDROID_LOG_DEBUG, "WEIMINN_PROJECT", msg)
7
8 int main(int argc, char *argv[]) {
9     ALOG("STARTING WEIMINN_PROJECT");
10
11     int n = 0;
12     while (true) {
13         sleep(DELAY_SECS);
14         n++;
15
16         ALOG("TESTING");
17     }
18 }
```

And create Soong build file, Android.bp in the same folder:

```

1 cc_binary {
2     name: "weiminn_daemon",
3     relative_install_path: "hw",
4     init_rc: ["init.weiminn.rc"],
5     header_libs: [
6         "liblog_headers",
7     ],
8     srcs: [
9         "weiminn.cpp"
10    ],
11     shared_libs: [
12         "liblog",
13         "libcutils",
14    ],
15     static_libs: [
16    ],
17     vendor: true,
18     proprietary: true,
19 }
```

Add wei_rudi_daemon to the PRODUCT_PACKAGES+=\ attribute of AOSP/device/generic/goldfish/vendor.mk

Add the startup script below to the end of the init.rc file:

```

1 service wei_rudi_daemon /vendor/bin/hw/wei_rudi_daemon
2     class main
3     user system
4     group system
5     oneshot
```

But the service cannot start due to SE Policy not being implemented for the service yet:

```

1 11-21 02:13:54.242 1862 1862 W cp : type=1400 audit(0.0:231): avc: denied {
   setattr } for path="/vendor/bin/hw/wei_rudi_daemon" dev="dm-3" ino=110
   scontext=u:r:shell:s0 tcontext=u:object_r:vendor_file:s0 tclass=file
   permissive=0

```

1.7 November 21, 2023

1.7.1 Fixing SELinux Policy to Start System Service

Add seclabel to the startup script:

```

1 service weiminn_daemon /vendor/bin/hw/weiminn_daemon
2   class main
3   user system
4   group system
5   seclabel u:r:ueventd:s0

```

Add start weiminn_daemon under on early-init right after start ueventd.

Got a new permission denied error this time:

```

1 11-21 09:45:47.732 0 0 E init : cannot execv('/vendor/bin/hw/weiminn_daemon'). See
   the 'Debugging init' section of init's README.md for tips: Permission denied
2 11-21 09:45:47.733 0 0 I init : Service 'weiminn_daemon' (pid 1402) exited with
   status 127
3 11-21 09:45:47.733 0 0 I init : Sending signal 9 to service 'weiminn_daemon' (pid
   1402) process group...

```

Adding device/generic/goldfish/sepolicy/x86/weiminn.te with following contents:

```

1 type weiminn, domain;
2 permissive weiminn;
3 type weiminn_exec, vendor_file_type, exec_type, file_type;
4
5 init_daemon_domain(weiminn)

```

and changed seclabel of the startup script to seclabel u:r:weiminn_exec:s0. And it reverts to the previous error:

```

1 11-21 10:19:43.865 0 0 I init : starting service 'weiminn_daemon',...
2 11-21 10:19:43.866 0 0 F init : cannot setexeccon('u:r:weiminn_exec:s0') for
   weiminn_daemon: Invalid argument
3 11-21 10:19:43.867 0 0 I init : Service 'weiminn_daemon' (pid 1827) exited with
   status 6
4 11-21 10:19:43.868 0 0 I init : Sending signal 9 to service 'weiminn_daemon' (pid
   1827) process group...
5 11-21 10:19:43.868 0 0 I libprocessgroup : Successfully killed process cgroup uid
   1000 pid 1827 in 0ms

```

1.7.2 Memory Dump

Try on Android Emulator

Set up emulator environment by appending the following inside `~/.bashrc`:

```

1 export ANDROID_SDK_ROOT=~/Android/Sdk
2 export ANDROID_HOME=~/Android/Sdk
3 export ANDROID_AVD_HOME=~/android/avd
4
5 PATH=$PATH:$ANDROID_SDK_ROOT/emulator

```

Register the path:

```
1 source ~/.bashrc
```

Create new AVD, Pixel 7 Pro with Tramisu (Android 13 with Google APIs, not Google Play), via Android Studio.

Run emulator using command:

```
1 emulator -avd Pixel_7_Pro_API_33
```

Load APK into emulator:

```

1 adb devices # to get id of running instance
2 adb root
3 adb install -r de.drmmaxnix.birthdaycountdown.apk

```

Clone Fridump⁵:

```

1 git clone https://github.com/Nightbringer21/fridump
2 code fridump
3 pip install frida frida-tools

```

Download and Run Fridump dependencies:

```

1 git clone https://github.com/Nightbringer21/fridump
2 code fridump
3 adb push frida-server /data/local/tmp
4 sudo sysctl kernel.yama.ptrace_scope=0 && frida-ps -D emulator-5554
5 frida-ps -D emulator-5554 | grep Birth # to get process ID of the Birthday app
6
7 # inside emulator shell
8 ./data/local/tmp/frida-server

```

Found out that Fridump doesn't support the latest Android 13 and API 34, so download Android 10 with API 29 with Pixel 3a.

Run Fridump:

```
1 python fridump.py -U -s Birthday\ Countdown
```

and it works now!

1.8 November 22, 2023

1.8.1 C++ Data Structure

Sales_item Struct

The data structure does not support any operations⁶ any requires the user to implement the operations themselves:

⁵<https://pentestcorner.com/introduction-to-fridump>

⁶Basically, class with only attributes, and no methods.

```

1 struct Sales_data {
2     std::string bookNo;
3     unsigned units_sold = 0;
4     double revenue = 0.0;
5 };

```

The names defined inside the class must be unique within the class but can reuse names defined outside the class. We define data members the same way that we define normal variables: We specify a base type followed by a list of one or more declarators.

The close curly that ends the class body must be followed by a semicolon. The semicolon is needed because we can define variables after the class body:

```

1 struct Sales_data { /* ... */ } accum, trans, *salesptr;
2 // equivalent, but better way to define these objects
3 struct Sales_data { /* ... */}; Sales_data accum, trans, *salesptr;

```

The semicolon marks the end of the (usually empty) list of declarators. Ordinarily, it is a bad idea to define an object as part of a class definition. Doing so obscures the code by combining the definitions of two different entities—the class and a variable—in a single statement.

We use the dot operator (.) to read into the member attributes of the object:

```

1 Sales_data data1, data2;
2 double price = 0; // price per book, used to calculate total revenue
3
4 // read the first transaction: ISBN, number of books sold, price per book
5 std::cin >> data1.bookNo >> data1.units_sold >> price;
6
7 // calculate total revenue from price and units_sold
8 data1.revenue = data1.units_sold * price;
9
10 // read the second transaction
11 std::cin >> data2.bookNo >> data2.units_sold >> price;
12 data2.revenue = data2.units_sold * price;

```

Preprocessor

In order to ensure that the class definition is the same in each file, classes are usually defined in header files. Typically, classes are stored in headers whose name derives from the name of the class. Thus, we will define our `Sales_data` class in a header file named `Sales_data.h`.

Headers often need to use facilities from other headers. For example, because our `Sales_data` class has a string member, `Sales_data.h` must `#include` the `string` header. As we've seen, programs that use `Sales_data` also need to include the `string` header in order to use the `bookNo` member. As a result, programs that use `Sales_data` will include the `string` header twice: once directly and once as a side effect of including `Sales_data.h`. Because a header might be included more than once, we need to write our headers in a way that is safe even if the header is included multiple times.

The **preprocessor** is a program that runs before the compiler and changes the source text of our programs. Our programs already rely on one preprocessor facility, `#include`. When the preprocessor sees a `#include`, it replaces the `#include` with the contents of the specified header.

Header guards can be defined using the preprocessor. Preprocessor variables have one of two states: defined and not defined. `#define` directive takes a name as a preprocessor variable. `#ifdef` is true if the variable has been defined, and `#ifndef` is true if the variable has not been defined. If the test is true, then everything following the `#ifdef` or `#ifndef` is processed up to the matching `#endif`:

```

1 #ifndef SALES_DATA_H
2
3 #define SALES_DATA_H
4 #include <string>
5
6 struct Sales_data {
7     std::string bookNo;
8     unsigned units_sold = 0;
9     double revenue = 0.0;
10 };
11
12 #endif

```

The first time `Sales_data.h` is included, the `#ifndef` test will succeed. The preprocessor will process the lines following `#ifndef` up to the `#endif`. As a result, the preprocessor variable `SALES_DATA_H` will be defined and the contents of `Sales_data.h` will be copied into our program. If we include `Sales_data.h` later on in the same file, the `#ifndef` directive will be false. The lines between it and the `#endif` directive will be ignored.

Preprocessor variables, including names of header guards, must be unique throughout the program. Typically we ensure uniqueness by basing the guard's name on the name of a class in the header. To avoid name clashes with other entities in our programs, preprocessor variables usually are written in all uppercase.

1.9 November 23, 2023

1.9.1 C++ Types and Data Structures

Aliases

Traditionally, we use `typedef` for synonym for another type:

```

1 typedef double wages; // wages is a synonym for double
2 typedef wages base, *p; // base is a synonym for double, p for double*

```

The new standard introduced a second way to define **alias declaration** type alias:

```
1 using SI = Sales_item; // SI is a synonym for Sales_item
```

It is also possible to declare "pointer to" alias:

```

1 typedef char *pstring;
2 const pstring cstr = 0; // equivalent to char *const cstr = 0;
3 const pstring *ps; // equivalent to const char *ps;

```

auto Type Specifier

We can let the compiler deduce the type from the initializer for us by using the **auto** type specifier:

```

1 // the type of item is deduced from the type of the result of adding val1 and val2
2 auto item = val1 + val2; // item initialized to the result of val1+val2

```

The initializer for all the variables must have types that are consistent with each other:

```

1 auto i = 0, *p = &i; // ok: i is int and p is a pointer to int
2 auto sz = 0, pi = 3.14; // error: inconsistent types for sz and pi

```

auto ignores top-level **consts**, and only low-level initializer are usually kept:

```

1 auto i = 0, *p = &i; // ok: i is int and p is a pointer to int
2 auto sz = 0, pi = 3.14; // error: inconsistent types for sz and pi

```

1.10 November 25, 2023

1.10.1 Android Memory Dump

Heap Dump

```

1 adb shell am dumpheap <PID> <HEAP-DUMP-FILE-PATH>
2 adb shell cat <HEAP-DUMP-FILE-PATH> > <LOCAL-FILE-PATH>
3 strings <LOCAL-FILE-PATH> <LOCAL-FILE-PATH-FOR-STRINGS>
4
5 pip install objection
6 frida-ps -Uai
7 # objection -g de.drmaxnix.birthdaycountdown explore
8 objection -g bloodpressure.bpdiary explore
9
10 android hooking list classes #List all loaded classes, As the target application
    gets usedmore, this command will return more classes.
11
12 android hooking search classes bloodpressure.bpdiary
13 android hooking search methods bloodpressure.bpdiary recordDbActivity
14
15 android hooking watch class bloodpressure.bpdiary.recordDbActivity --dump-args --
    dump-return
16
17 android hooking watch class_method bloodpressure.bpdiary.recordDbActivity.
    showWeight --dump-args --dump-backtrace --dump-return

```

1.11 November 27, 2023

1.11.1 Android Profile Guided Compilation

7 8

Baseline Profiles

```
1 adb shell cmd package dump-profiles bloodpressure.bldiary
2 adb shell cat /data/misc/profman/bloodpressure.bldiary-primary.prof.txt >
   bldiary_profile.txt
```

Perfetto

Simpleperf

9

```
1 git clone https://android.googlesource.com/platform/system/extras
2 cd extras/simpleperf/demo/scripts/
3 python3 app_profiler.py -p simpleperf.example.java
4 python3 report_html.py --add_source_code --source_dirs ../demo --add_disassembly
5 sudo apt-get install python3-tk
6 ./report.py

1 python3 app_profiler.py -p bloodpressure.bldiary
2 adb shell /data/local/tmp/simpleperf record -o /data/local/tmp/perf.data -e task-
   clock:u -f 1000 -g --duration 10 --log info --app bloodpressure.bldiary
```

Profcollect

10

This is only supported by Coresight ETM-enabled ARM devices, so emulator doesn't work.

Inside ADB shell:

```
1 device_config put profcollect_native_boot enabled true
2 setprop persist.device_config.profcollect_native_boot.collection_interval 60
3 setprop persist.device_config.profcollect_native_boot.sampling_period 1000
4 setprop persist.device_config.profcollect_native_boot.max_trace_limit 53687091200
5 setprop persist.device_config.profcollect_native_boot.enabled true
6 setprop ctl.stop profcollectd
7 setprop ctl.start profcollectd
8 ps -e | grep profcollectd
```

⁷<https://developer.android.com/games/agde/pgo-overview>

⁸<https://newandroidbook.com/files/ArtOfDalvik.pdf>

⁹<https://android.googlesource.com/platform/system/extras/+/main/simpleperf/demo/README.md>

¹⁰<https://android.googlesource.com/platform/system/extras/+/master/profcollectd/>

1.12 November 29, 2023

1.12.1 Android OAT Dump Parser

The format for OAT files is described in AOSP/art/dex2oat/linker/oat_writer.h.

.bss¹¹ is the portion of an object file, executable, or assembly language code that contains statically allocated variables that are declared but not have been assigned a value yet.

```

1 import argparse
2
3 # helper function
4 def findLineWith(arr, s):
5     for i in range(len(arr)):
6         if s in arr[i]:
7             return i
8
9 def parse_method(method_arr):
10    method_sig_raw = method_arr.pop(0)
11    method_sig_raw_ = method_sig_raw[0: method_sig_raw.index(',')+1]
12    method_sig_raw__ = method_sig_raw_.split(':')[1:]
13    method_sig = method_sig_raw__[0].strip()
14
15    method_arr.pop(0) # discard "DEX CODE:"
16    endOfDex = findLineWith(method_arr, "OatMethodOffsets")
17    dexCode = method_arr[0: endOfDex]
18
19    startOfOat = findLineWith(method_arr, "CODE:")
20    nativeCode = method_arr[startOfOat+1:] # Exclude "CODE: "
21
22    parsed_method = {'method_sig': method_sig, 'dex': dexCode, 'native':
23                     nativeCode}
24
25    # print("Parsed", method_sig)
26
27    return parsed_method
28
29 def parse_type(type_arr):
30    method_indexes = [i for i in range(len(type_arr)) if 'method_idx' in type_arr[
31        i]]
32
33    if len(method_indexes) > 0:
34        method_indexes.append(len(type_arr))
35    methods_raw = [type_arr[method_indexes[i]:method_indexes[i+1]] for i in range(
36        0, len(method_indexes)-1)]
37
38    methods = []
39    for mraw in methods_raw:
40        methods.append(parse_method(mraw))
41
42    return {'type_name': type_arr[0].split()[1], 'methods': methods}
43
44 def load_oat_dump(p):
45     f = open(p)
46     lines = f.readlines()
47
48

```

¹¹<https://en.wikipedia.org/wiki/.bss>

```

44     # determine the start and end of DEX code
45     startOfOatDex = lines.index('OatDexFile:\n')
46     endOfOatDex = lines.index('OATFILESTATS:\n')
47     OatDex = lines[startOfOatDex:endOfOatDex]
48
49     # extract the classes and types
50     type_indexes = [i for i in range(len(OatDex)) if 'type_idx' in OatDex[i]]
51     type_indexes.append(len(OatDex))
52     all_types = [OatDex[type_indexes[i]:type_indexes[i+1]] for i in range(0, len(
53         type_indexes)-1)]
53
54     all_methods = []
55     all_methods_dict = {}
56
57     for t in all_types:
58         parsed = parse_type(t)
59         all_methods += parsed['methods']
60
61         for m in parsed['methods']:
62             all_methods_dict[m['method_sig']] = {}
63             all_methods_dict[m['method_sig']]['dex'] = m['dex']
64             all_methods_dict[m['method_sig']]['native'] = m['native']
65
66     # return all_methods
67     return all_methods_dict
68
69 def load_schema(p):
70     f = open(p)
71     lines = f.readlines()
72
73     schema = []
74     for l in lines:
75         split = l.split('|||')
76         schema.append(split[1][:-1]) # -1 to remove \n character
77         # print("Added schema", schema[-1])
78     return schema
79
80 parser = argparse.ArgumentParser(description='OATDebloationChecker')
81 parser.add_argument('--oat', type=str, help='OATDumpTxtfile')
82 parser.add_argument('--schema', type=str, help='DebloatingSchemafile')
83
84 try:
85     args = parser.parse_args()
86     # print("Options:", args.OAT, args.schema)
87
88     # Load Debloated OAT Dump
89     print("DebloatedOATDump:", args.oat)
90     normal_methods = load_oat_dump(args.oat)
91     # normal_methods = load_oat_dump('prelim_results/com.storiestime/10
92     # min_normal_com.storiestime_oat_dump.txt')
93
94     # Load Schema
95     print("SchemaFile:", args.schema)
96     schema = load_schema(args.schema)
97     # schema = load_schema('Money_script/success_schema/169_com.
98     # storiestime_removed_methods.txt')
99
98     # Check if debloated
99     for m in schema:

```

```
100     retrieve = normal_methods[m]
101     if 'NO_CODE!' in retrieve['native'][0]:
102         print(m, 'is_debloated!')
103     else:
104         print(m, 'is_not_debloated!')
105
106     # print('test')
107 except Exception as e:
108     print("Exception!", str(e))
```


Chapter 2

December

2.1 December 01, 2023

2.1.1 C++ Strings and Vectors

Namespace using Declarations

The scope operator (::) says that the compiler should look in the scope of the left-hand operand for the name of the right-hand operand. using declaration lets us use a name from a namespace without qualifying the name with namespace:: prefix as below:

```
1 #include <iostream>
2
3 // when we use the name cin, we get the one from the namespace std
4 using namespace::name;
5
6 int main(){
7     int i;
8     cin >> i;
9     cout << i;
10    std::cout << i;
11    return 0;
12 }
```

Headers should NOT include using declarations, as the contents of a header are copied into the including program's text. As a result, a program that didn't intend to use the specified library name might encounter unexpected name conflicts.

string Type

If you include a header that includes <string> you may not have to do so explicitly. However, it is bad practice to count on this, and well-written headers include guards against multiple inclusion, so assuming you're using well-written header files, there is

no harm in including a header that was included via a previous include.¹:

```
1 #include <string> // You need to include the <string> header to use std::string.
2 using std::string;
```

Most common ways to initialize strings:

```
1 // direct initialization
2 string s1; // default initialization; s1 is the empty string
3 string s4(10, 'c'); // s4 is cccccccccc
4 string s6("hiya");
5
6 // copy initialization
7 string s2 = s1; // s2 is a copy of s1
8 string s3 = "hiya"; // equivalent to string s3("hiya");
9 string s8 = string(10, 'c') // copying initialization; s8 is cccccccccc
```

When we initialize a variable using `=`, we are asking the compiler to copy initialize the object by copying the initializer on the right-hand side into the object being created. When we omit the `=`, we use direct initialization.

string Operations:

Statement	Description
<code>os << s</code>	Writes <code>s</code> onto output stream <code>os</code> . Return <code>os</code> .
<code>is >> s</code>	Reads whitespace-separated string from <code>is</code> to <code>s</code> . Return <code>is</code> .
<code>getline(is, s)</code>	Reads a line of input from <code>is</code> into <code>s</code> . Returns <code>is</code> .
<code>s.empty()</code>	Returns <code>true</code> if <code>s</code> is empty; otherwise return <code>false</code> .
<code>s.size()</code>	Returns the number of characters in <code>s</code> .
<code>s[n]</code>	Returns a reference to the <code>char</code> at position <code>n</code> in <code>s</code> .
<code>s1 + s2</code>	Returns a <code>string</code> that is the concatenation of <code>s1</code> and <code>s2</code> .
<code>s1 += s2</code>	Equivalent to <code>s1 = s1 + s2</code> .
<code>s1 = s2</code>	Replaces characters in <code>s1</code> with a copy of <code>s2</code> .
<code>s1 == s2</code> and <code>s1 != s2</code>	The strings in <code>s1</code> and <code>s2</code> are equal if they contain the same characters. The equality is case-sensitive.
<code><, <=, >, >=</code>	Comparisons are case-sensitive and use dictionary ordering.

Table 2.1: string Operations

To check individual character:

Function	Description
<code>isalnum(c)</code>	true if <code>c</code> is a letter or a digit.
<code>isalpha(c)</code>	true if <code>c</code> is a letter.
<code>iscntrl(c)</code>	true if <code>c</code> is a control character.

¹<https://stackoverflow.com/a/73640984>

Function	Description
digit(c)	true if c is a digit.
isgraph(c)	true if c is not a space but is printable.
islower(c)	true if c is a lowercase letter.
isprint(c)	true if c is a printable character (i.e., a space or a character that has a visible representation).
ispunct(c)	true if c is a punctuation character.
isspace(c)	true if c is whitespace (i.e., a space, tab, vertical tab, return, newline, or formfeed).
isupper(c)	true if c is an uppercase letter.
isxdigit(c)	true if c is a hexadecimal digit.
tolower(c)	If c is an uppercase letter, returns its lowercase equivalent; otherwise returns c unchanged.
toupper(c)	If c is an lower letter, returns its uppercase equivalent; otherwise returns c unchanged.

Table 2.2: ctype Functions

Reading an Unknown Number of strings

If the stream is valid - it hasn't hit end-of-file or encountered an invalid input - then the body of while is executed:

```

1 int main() {
2     string word;
3     while (cin >> word) // read until end-of-file
4         cout << word << endl; // write each word followed by a new line
5
6     // Reads the given stream up to and including the first newline
7     string line; // read input a line at a time until end-of-file
8     while (getline(cin, line))
9         // The newline that causes getline to return is discarded; the newline is
10        // not stored in the string.
11        cout << line << endl;
12
13     while (getline(cin, line))
14         // Only print lines that are not empty
15         if (!line.empty())
16             cout << line << endl;
17
18 }
```

Library strings and String Literals

When we mix strings and string or character literals, at least one operand to each + operator must be of string type:

```

1 string s4 = s1 + ","; // ok: adding a string and a literal
2 string s5 = "hello" + ","; // error: no string operand
3 string s6 = s1 + "," + "world"; // ok: each + has a string operand
4 string s7 = "hello" + "," + s2; // error: can't add string literals

```

For compatibility reasons with C, string literals are NOT standard library strings. It is important to remember that these types differ when you use string literals and library strings.

Characters in strings

string expression represent a sequence of characters, and to traverse every character, we can use a range for that follows the syntax:

```

1 for (declaration: expression)
2     statement

```

A simple example:

```

1 string str("some\u00f6string");
2 // print the characters in str one character to a line
3 for (auto c : str) // for every char in str
4     cout << c << endl; // print the current character followed by a newline

```

We use auto to let compiler deduce the type of c, which in this case will be char.

If we want to change the value of the character in a string, we must define the loop variable as a reference type:

```

1 string str("some\u00f6string");
2 // convert s to uppercase
3 for (auto &c : str) // for every char ref in str
4     c = toupper(s); // c is a reference, so the assignment changes the char in s

```

The subscript operator (the [] operator) takes a `string::size_type` value that denotes the position of the character we want to access. The operator returns a reference to the character at the given position:

```

1 string str("some\u00f6string");
2 if (!s. empty())           // make sure there's a character to print
3     cout << s[0] << endl;   // print the first character in s

```

To iterate using subscript:

```

1 // process characters in s until we run out of characters or we hit a whitespace
2 for (
3     decltype(s.size()) index = 0;
4     index != s.size() && !isspace(s[index]); // the operator yields true if both
        operands are true
5     ++index)
6     s[index] = toupper(s[index]); // capitalize the current character

```

vector Type

A vector is a collection² of objects, all of which have the same type, and each of which has an associated index that gives access to that object. Below is the headers to use a vector:

```
1 #include <vector>
2 user std::vector;
```

vector is not a class/type but a class template, which is a set of instructions for the compiler for generating classes/types, a process called instantiation. To specify what kind of class (what type of object we want the vector to hold) we want to instantiate, we supply additional information inside a pair of angle brackets following the template's name:

```
1 vector<int> ivec; // ivec holds objects of type
2 int vector<Sales_item> Sales_vec; // holds Sales_items
3 vector<vector<string>> file; // vector whose elements are vectors
```

Here, `vector<int>`, `vector<Sales_item>`, and `vector<vector<string>>` are the types generated types by the compiler.

Because references are not objects, we cannot have a vector of references.

Defining and Initializing vectors

The most common way of Initializing a vector is to initialize an empty vector. We can also perform direct and copy initialization, but the objects must be the same type:

```
1 vector<string> svec; // default initialization; svec has no elements
2
3 // direct and copy initialization
4 vector<int> ivec2(ivec); // copy elements of ivec into ivec2
5 vector<int> ivec3 = ivec; // copy elements of ivec into ivec3
6 vector<string> svec(ivec2); // error: svec holds strings, not ints
7
8 // list initialization
9 vector<string> articles = {"a", "an", "the"};
10 vector<string> articles2{"a", "an", "the"};
11 vector<string> articles3("a", "an", "the"); // error
12
13 vector<int> ivec(10, -1); // ten int elements, each initialized to -1
14 vector<string> svec(10, "hi!"); // ten strings; each element is "hi!"
15
16 vector<int> ivec(10); // ten elements, each initialized to 0
17 vector<string> svec(10); // ten elements, each an empty string
```

Some classes require that we always supply an explicit initializer, and cannot be default initialized, in which case, we must supply the initial value/

²Often referred to as container because it "contains" other objects.

vector Operations

Two vectors are equal if they have the same number of elements, and if the corresponding elements all have the same value. If the vectors have differing sizes, but the elements that are in common are equal, then the vector with fewer elements is less than the one with more elements. If the elements have differing values, then the relationship between the vectors is determined by the relationship between the first elements that differ. We can compare two vectors only if we can compare the element in those vectors.

Methods	Description
<code>v.empty()</code>	Returns true if <code>v</code> is empty; otherwise returns false.
<code>v.size()</code>	Returns the number of elements in <code>v</code> .
<code>v.push_back(t)</code>	Adds an element with value <code>t</code> to the end of <code>v</code> .
<code>v[n]</code>	Returns a reference to the element at position <code>n</code> in <code>v</code> .
<code>v1 = v2</code>	Replaces the elements in <code>v1</code> with a copy of the elements in <code>v2</code> .
<code>v1 = {a, b, c, ...}</code>	Replaces the elements in <code>v1</code> with a copy of the elements in the comma-separated list.
<code>v1 == v2</code> and <code>v1 != v2</code>	<code>v1</code> and <code>v2</code> are equal if they have the same number of elements and each element in <code>v1</code> is equal to corresponding element in <code>v2</code> .
<code><, <=, >, >=</code>	Have their normal meanings using dictionary ordering.

Table 2.3: vector Methods

As with strings, subscript for vector start at 0; the type of a subscript is the corresponding `size_type`; and we can write to the element returned by the subscript operator.

Subscripting a vector does NOT add elements:

```

1 vector<int> ivec; // empty vector
2 for (decltype(ivec.size()) ix = 0; ix != 10; ++ix)
3     ivec[ix] = ix; // disaster: ivec has no elements

```

It is an error to subscript an element that doesn't exist, but it is an error that the compiler is unlikely to detect. Instead, the value we get at run time is undefined³. A good way to ensure that subscripts are in range is to avoid subscripting altogether by using a range for whenever possible.

³Buffer overflow errors are the result of subscripting elements that don't exist. Such bugs are the most common cause of security problems in PC and other applications.

2.2 December 03, 2023

2.2.1 C++ Iterator, and Arrays

Iterators

Iterators are more general mechanism than subscript operators. All of the library containers have iterators, but only a few of them support the subscript operator. Like pointers, iterators give us indirect access to an object.

```
1 // the compiler determines the type of b and e;
2 // b denotes the first element and e denotes one past the last element in v
3 auto b = v.begin(), e = v.end(); // b and e have the same type
```

The begin member returns an iterator that denotes the first element (if there is one). The end member returns an iterator positioned "one past the end" of the associated container (or string), also referred to as the off-the-end iterator. If the container is empty, begin returns the same iterator as the one returned by end.

We do not know (or need to care about) the precise type that an iterator has, so we use `auto` to define `b` and `e`.

We compare two valid iterators using `==` or `!=`. Iterators are equal if they denote the same element or if they are both off-the-end iterators for the same container. Otherwise, they are unequal.

Like pointers, we can dereference a valid iterator to obtain the element denoted by an iterator. Dereferencing an invalid iterator or an off-the-end iterator has undefined behavior:

```
1 string s("some\u00f6string");
2 if (s.begin() != s.end()) { // make sure s is not empty
3     auto it = s.begin(); // it denotes the first character in s
4     *it = toupper(*it); // make that character uppercase
5 }
```

Iterators also support a few other operations:

Methods	Description
<code>*iter</code>	Returns a reference to the element denoted by the iterator <code>iter</code> .
<code>iter->mem</code>	Dereferences <code>iter</code> and fetches the member named <code>mem</code> from the underlying element. Equivalent to <code>(*iter).mem</code> .
<code>++iter</code>	Increments <code>iter</code> to refer to the next element in the container.
<code>-iter</code>	Decrements <code>iter</code> to refer to the previous element in the container.
<code>iter1 == iter2</code> and <code>iter1 != iter2</code>	Compares two iterators for equality. Two iterators are equal if they denote the same element or if they are off-the-end iterator for the same container.

Table 2.4: Iterator operations

The increment (++) operator to move from one element to the next:

```

1 string s("someustring");
2 for (auto it = s.begin(); it != s.end() && !isspace(*it); ++it) {
3     *it = toupper(*it); // make that character uppercase
4 }
```

Because the iterator returned by `end` does not denote an element, it may not be incremented or dereferenced.

When we need to read but not write to an object, we ask specifically for `const_iterator` type:

```

1 vector<int> v;
2 auto b = v.cbegin(); // b has type vector<int>::const_iterator
3 auto e = v.cend(); // e has type vector<int>::const_iterator
```

Regardless of whether the container is `const`, they return a `const_iterator`.

Iterators for `string` and `vector` support additional operations that can move an iterator multiple elements at a time, often referred to as iterator arithmetic:

Iter. Arithmetic	Description
<code>iter + n</code> and <code>iter - n</code>	Adding (subtracting) an integral value <code>n</code> to (from) an iterator yields an iterator that many elements forward (or backward) within the container. The resulting iterator must denote elements in, or one past the end of, the same container.
<code>iter += n</code> and <code>iter -= n</code>	Compound-assignment for iterator addition and subtraction. Assigns to <code>iter1</code> the value of adding <code>n</code> to, or subtracting <code>n</code> from, <code>iter1</code> .
<code>iter1 - iter2</code>	Subtracting two iterators yields the number that when added to the right-hand iterator yields the left-hand iterator. The iterators must denote elements in, or one past the end of, the same container.
<code>>,>=,<,<=</code>	Relational operators on iterators. One iterator is less than another if it refers to an element that appears in the container before the one referred to by the other iterator. The iterators must denote elements in, or one past the end of, the same container.

Table 2.5: Iterator Arithmetics

A classic algorithm that uses iterator arithmetic is binary search:

```

1 vector<int> v;
2 // text must be sorted
3 // beg and end will denote the range we're searching
4 auto beg = text.begin(), end = text.end();
```

```

5 auto mid = text.begin() + (end - beg)/2; // original midpoint
6
7 // while there are still elements to look at and we haven't yet found sought
8 while (mid != end && *mid != sought) {
9     if (sought < *mid) // is the element we want in the first half?
10        end = mid; // if so, adjust the range to ignore the second half
11    else // the element we want is in the second half
12        beg = mid + 1; // start looking with the element just after mid
13    mid = beg + (end - beg)/2; // new midpoint
14 }

```

Arrays

Similar to library `vector` type, an array is a container of unnamed objects of a single type that we access by position. Unlike a `vector`, arrays have fixed size; we cannot add elements to an array, in order to attain better runtime performance for specialize applications at the cost of flexibility.

An array declarator has the form `a[d]`, where `a` is the name being defined and `d` is the dimension of the array which specifies the number of elements and must be greater than zero. The dimension must be known at compile time, which means that the dimension must be a `constexpr`:

```

1 unsigned cnt = 42; // not a constant expression
2 constexpr unsigned sz = 42; // constant expression
3 int arr[10]; // array of ten ints
4 int *parr[sz]; // array of 42 pointers to int
5 string bad[cnt]; // error: cnt is not a constant expression
6 string strs[get_size()]; // ok if get_size is constexpr, error otherwise

```

By default, the elements in an array are default initialized. As with `vector`, arrays hold objects. Thus, there are no arrays of references.

We can list initialize an array which allow us to omit the dimension as the compiler infers it from the number of initializers. If we specify, the number of initializers must not exceed the specified size:

```

1 const unsigned sz = 3;
2 int ia1[sz] = {0,1,2}; // array of three ints with values 0, 1, 2
3 int a2[] = {0, 1, 2}; // an array of dimension 3
4 int a3[5] = {0, 1, 2}; // equivalent to a3[] = {0, 1, 2, 0, 0}
5 string a4[3] = {"hi", "bye"}; // same as a4[] = {"hi", "bye", ""}
6 int a5[2] = {0,1,2}; // error: too many initializers

```

Character arrays can also be initialized from a string literals. It's important to remember that string literals end with a null character:

```

1 char a1[] = {'C', '+', '+'}; // list initialization, no null
2 char a2[] = {'C', '+', '+', '\0'}; // list initialization, explicit null
3 char a3[] = "C++"; // null terminator added automatically
4 const char a4[6] = "Daniel"; // error: no space for the null!

```

We cannot initialization an array as a copy of another array, nor is it legal to assign one array to another⁴:

```

1 int a[] = {0, 1, 2}; // array of three ints
2 int a2[] = a; // error: cannot initialize one array with another
3 a2 = a; // error: cannot assign one array to another

```

Defining arrays that hold pointers is fairly straightforward, defining a pointer or reference to an array is a bit more complicated:

```

1 int *ptrs[10]; // ptrs is an array of ten pointers to int
2 int &refs[10] = /* ? */; // error: no arrays of references
3 int (*Parray)[10] = &arr; // Parray points to an array of ten ints
4 int (&arrRef)[10] = arr; // arrRef refers to an array of ten ints

```

The parentheses around `*Parray` means that `Parray` is a pointer. Looking right, we see that `Parray` points to an array of size 10. Looking left, we see taht the elements in the array are `ints..` Thus, `Parray` is a pointer to an array of ten `ints`. Similarly, `&arrRef` says that `arrRef` is a reference.

We can use range for or the subscript operator to access elements of an array:

```

1 // count the number of grades by clusters of ten: 0--9, 10--19, . . . 90--99, 100
2 unsigned scores[11] = {}; // 11 buckets, all value initialized to 0
3 unsigned grade;
4 while (cin >> grade) {
5     if (grade <= 100)
6         ++scores[grade/10]; // increment the counter for the current cluster
7 }
8
9 for (auto i : scores) // for each counter in scores
10    cout << i << " "; // print the value of that counter
11 cout << endl;

```

We have to use a variable to have type `size_t` (defined in `cstdint` header) which is a machine-specific unsigned type that is guaranteed to be large enough to hold the size of any object in memory.

The most common source of security problems are buffer overflow bugs. Such bugs occur when a program fails to check a subscript and mistakenly uses memory outside the range of an array or similar data structure. Nothing stops a program from stepping across an array boundary except careful attention to detail and thorough testing of the code.

We obtain a pointer to an array element by taking the address of that element:

```

1 string nums[] = {"one", "two", "three"}; // array of strings
2 string *p = &nums[0]; // p points to the first element in nums
3 string *p2 = nums; // equivalent to p2=&nums[0]

```

When we use an object of array type, we are really using a pointer to the first element in that array, as the compiler automatically substitutes a pointer to the first element.

When we use an array as an initializer for a variable defined using `auto`, the deduced type is a pointer, not an array:

⁴Some compilers allow array assignment as a compiler extension. It is usually a good idea to avoid using nonstandard features. Programs that use such features, will not work with a different compiler.

```

1 int ia[] = {0,1,2,3,4,5,6,7,8,9}; // ia is an array of ten ints
2 auto ia2(ia); // ia2 is an int* that points to the first element in ia
3 ia2 = 42; // error: ia2 is a pointer, and we can't assign an int to a pointer

```

Although ia is an array of ten ints, when we use ia as an initializer, the compiler treats that initialization as if we had written:

```
1 auto ia2(&ia[0]); // now it's clear that ia2 has type int*
```

This conversion does not happen when we use decltype:

```

1 // ia3 is an array of ten ints
2 decltype(ia) ia3 = {0,1,2,3,4,5,6,7,8,9};
3 ia3 = p; // error: can't assign an int* to an array
4 ia3[4] = i; // ok: assigns the value of i to an element in ia3

```

Pointers to array elements support the same operations as iterators on vectors or strings:

```

1 // ia3 is an array of ten ints
2 int arr[] = {0,1,2,3,4,5,6,7,8,9};
3 int *p = arr; // p points to the first element in arr
4 ++p; // ppoints to arr[1]
5 int *e = &arr[10]; // pointer just past the last element in arr

```

arr has 10 elements, so the last element in arr is at index 9. Like the off-the-end iterator, off-the-end pointer does not point to an element. As a result, we may not dereference or increment an off-the-end pointer.

To be safer and less error-prone, we can use begin and end functions that act like similarly named container members. However, as arrays are not class types, these are not member functions, so they take an argument that is an array:

```

1 int ia[] = {0,1,2,3,4,5,6,7,8,9}; // ia is an array of ten ints
2 int *beg = begin(ia); // pointer to the first element in ia
3 int *last = end(ia); // pointer one past the last element in ia

```

Pointers that address array elements can use all iterator operations in Table 2.4 and Table 2.5. When we add an integral value to or from a pointer, the result is a new pointer. That new pointer points to the element the given number ahead of the original pointer:

```

1 constexpr size_t sz = 5;
2 int arr[sz] = {1,2,3,4,5};
3 int *ip = arr; // equivalent to int*ip=&arr[0]
4 int *ip2 = ip + 4; // ip2points to arr[4], the last element in arr
5
6 // ok: arr is converted to a pointer to its first element; ppoints one past the end
   of arr int *p = arr + sz; // use caution -- do not dereference! int *p2 = arr
   + 10; // error: arr has only 5 elements; p2 has undefined value

```

When we add sz to arr, the compiler converts arr to a pointer to the first element in arr. As a result, we can dereference the resulting pointer:

```

1 int ia[] = {0,2,4,6,8}; // array with 5 elements of type int
2 int last = *(ia + 4); // ok: initializes last to 8, the value of ia[4]
3 last = *ia + 4; // ok: last=4, equivalent to

```

As with iterators, subtracting two pointers gives us the distance between those pointers. The pointers must point to elements in the same array:

```
1 auto n = end(arr) - begin(arr); // n is 5, the number of elements in arr
```

The result of subtracting two pointers is a library type named `ptrdiff_t` which is a machine-specific type and is defined in the `cstddef` header.

When we subscript an array, we are subscripting a pointer to an element in that array:

```
1 int i = ia[2]; // ia is converted to a pointer to the first element in ia
2 // ia[2] fetches the element to which (ia+2) points
3 int *p = ia; // p points to the first element in ia
4 i = *(p + 2); // equivalent to i = ia[2]
5 int k = p[-2]; // p[-2] is the same element as ia[0]
```

Unlike subscripts for `vector` and `string`, the index of the built-in subscript operator is not an `unsigned` type.

Modern C++ programs should use vectors and iterators instead of built-in arrays and pointers, and use strings rather than C-style array-based character strings. Pointers are used for low-level manipulations and it is easy to make bookkeeping mistakes. Other problems arise because of the syntax, particularly the declaration syntax used with pointers.

Multidimensional Arrays

Multidimensional arrays in C++ are actually arrays of arrays:

```
1 int ia[3][4]; // array of size 3; each element is an array of ints of size 4
2 // array of size 10; each element is a 20-element array whose elements are arrays
3 // of 30 ints
4 int arr[10][20][30] = {0}; // initialize all elements to 0
5
6 int ia[3][4] = { // three elements; each element is an array of size 4
7     {0, 1, 2, 3}, // initializers for the row indexed by 0
8     {4, 5, 6, 7}, // initializers for the row indexed by 1
9     {8, 9, 10, 11} // initializers for the row indexed by 2
10 };
11 // equivalent initialization without the optional nested braces for each row
12 int ia[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
13
14 // explicitly initialize only element 0 in each row
15 int ia[3][4] = {{ 0 }, { 4 }, { 8 }};
16
17 // explicitly initialize row 0; the remaining elements are value initialized
18 int ix[3][4] = {0, 3, 6, 9};
19
20 // assigns the first element of arr to the last element in the last row of ia
21 ia[2][3] = arr[0][0][0];
22
23 int (&row)[4] = ia[1]; // binds row to the second four-element array in ia
```

2.3 December 04, 2023

2.3.1 C++ Expressions I

lvalues and rvalues

Every expression is either an rvalue or an lvalue. lvalues could stand on the left-hand side of an assignment where as rvalue could not. Roughly speaking, when we use an object as an rvalue, we use the object's value (its contents). When we use an object as an lvalue, we use the object's identity (its location in memory). We can use an lvalue when an rvalue is required, but we cannot use an rvalue when an lvalue (i.e., a location) is required. When we use an lvalue in place of an rvalue, the object's contents (its value) are used:

- Assignment requires a (non `const`) lvalue as its left-hand operand and yields its left-hand operand as an lvalue.
- The address-of operator requires an lvalue operand and returns a pointer to its operand as an rvalue.
- The built-in dereference and subscript operators and the iterator dereference and `string` and `vector` subscript operator all yield lvalues.
- The built-in and iterator increment and decrement operators require lvalue operands and the prefix versions also yield lvalues.

Lvalues and rvalues also differ when used with `decltype`. When we apply `decltype` to an expression, the result is a reference type if the expression yields an lvalue.

Arithmetic Operators

Division between integers returns an integer. If the quotient contains a fractional part, it is truncated toward zero:

```
1 int ival1 = 21/6; // ival1 is 3; result is truncated; remainder is discarded int
2 ival2 = 21/7; // ival2 is 3; no remainder; result is an integral value
```

For most operators, operands of type `bool` are promoted to `int`. In this case, the value of `b` is `true`, which promotes to the `int` value `1`. That (promoted) value is negated, yielding `-1`. The value `-1` is converted back to `bool` and used to initialize `b2`. This initializer is a nonzero value, which when converted to `bool` is `true`. Thus, the value of `b2` is `true`!

The operands to `%` must have integral type:

```

1 int ival = 42;
2 double dval = 3.14;
3
4 ival % 12; // ok: result is 6
5 ival % dval; // error: floating-point operand

```

2.4 December 05, 2023

2.4.1 C++ Expressions II

Logical and Relational Operators

The relational operators take operands of arithmetic or pointer type; the logical operators take operands of any type that can be converted to `bool`. The operands to those operators are `rvalues` and the result is an `rvalue`.

Associativity	Operator	Function	Use
Right	!	logical NOT	<code>!expr</code>
Left	<	less than	<code>expr < expr</code>
Left	<=	less than or equal	<code>expr <= expr</code>
Left	>	greater than	<code>expr > expr</code>
Left	>=	greater than or equal	<code>expr >= expr</code>
Left	==	equality	<code>expr == expr</code>
Left	!=	inequality	<code>expr != expr</code>
Left	&&	logical AND	<code>expr && expr</code>
Left		logical OR	<code>expr expr</code>

Table 2.6: Logical and Relational Operators

Because relational operators return `bools`, the result of chaining these operators together is likely to be surprising:

```

1 // oops! this condition compares k to the bool result of i<j
2 if(i<j<k) // true if k is greater than 1!

```

The compiler converts `val` to `bool`:

```

1 if (val) { /* ... */} // true if val is any nonzero value
2 if (!val) { /* ... */} // true if val is zero
3 if (val == true) { /* ... */} // true only if val is equal to 1!
4 if (val == 1) { /* ... */}

```

If `val` is not `bool`, then `true` is converted to the type of `val` before the `==` operator is applied.

Assignment Operators

The left-hand operand of an assignment operator must be a modifiable lvalue. For example, given:

```

1 int i = 0, j = 0, k = 0;      // initializations, not assignment
2 const int ci = i;            // initialization, not assignment
3 1024 = k;                  // error: literals are rvalues
4 i + j = k;                  // error: arithmetic expressions are rvalues
5 ci = k;                     // error: ci is a const(nonmodifiable) lvalue
6 k = 0;                      // result: type int, value 0
7 k = 3.14159;                // result: type int, value 3
8 k = {3.14};                 // error: narrowing conversion
9 vector<int> vi;           // initially empty
10 vi = {0,1,2,3,4,5,6,7,8,9}; // vi now has ten elements, values 0 through 9

```

Unlike the other binary operators, assignment is right associative. The right-most assignment, `jval = 0`, is the right-hand operand of the left-most assignment operator:

```

1 int ival, jval;
2 ival = jval = 0; // ok: each assigned 0
3 int ival, *pval; // ival is an int; pval is a pointer to int
4 ival = pval = 0; // error: cannot assign the value of a pointer to an int
5 string s1, s2;
6 s1 = s2 = "OK"; // string literal "OK" converted to string

```

Each object in a multiple assignment must have the same type as its right-hand neighbor or a type to which that neighbor can be converted.

Assignment often occur in conditions. Because assignment has relatively low precedence, we usually must parenthesize the assignment for the condition to work properly:

```

1 // a verbose and therefore more error-prone way to write this loop
2 int i = get_value(); // get the first value
3 while (i != 42) {
4     // do something . .
5     i = get_value();
6     // get remaining values
7
8 }
9
10 int i;
11 // a better way to write our loop---what the condition does is now clearer
12 while ((i = get_value()) != 42) {
13     // do something . .
14 }

```

Assignment Operators

The dot and arrow operators provide for member access. The dot operator fetches a member from an object from an object of class type; arrow is defined so that `ptr->mem` is a synonym for `(*ptr).mem`:

```

1 string s1 = "a\ustring", *p = &s1;
2 auto n = s1.size(); // run the sizemember of the strings1

```

```

3 n = p->size();      // equivalent to (*p).size()
4 n=(*p).size();      // run size on the object to which p points
5
6 // run the size member of p, then dereference the result!
7 *p.size(); // error: p is a pointer and has no member named size

```

Because dereference has a lower precedence than dot, we must parenthesize the dereference subexpression.

Conditional Operator

The conditional (the ?: operator) lets us embed simple if-else logic inside an expression:

```

1 // cond ? expr1: expr2;
2 string final grade = (grade < 60) ? "fail": "pass";

```

where `expr1` and `expr2` are expressions of the same type.

An incompletely parenthesized conditional operator in an output expression can have surprising results:

```

1 cout << ((grade < 60) ? "fail" : "pass"); // prints pass or fail
2 cout << (grade < 60) ? "fail" : "pass"; // prints 1 or 0!
3 cout << grade < 60 ? "fail" : "pass"; // error: compares cout to 60

```

The second expression uses the comparison between `grade` and 60 as the operand to the `<<` operator.

Bitwise Operator

Because there are no guarantees for how the sign bit is handled, it is strongly recommended to use `unsigned` types with the bitwise operators.

Operator	Function	Use
	bitwise NOT	<code>expr</code>
<code><<</code>	left shift	<code>expr1 << expr2</code>
<code>>></code>	right shift	<code>expr1 >> expr2</code>
<code>&</code>	bitwise AND	<code>expr1 & expr2</code>
<code>^</code>	bitwise XOR	<code>expr1 ^ expr2</code>
<code> </code>	bitwise OR	<code>expr1 expr2</code>

Table 2.7: Bitwise Operators

The built-in meaning of the shift operators is to perform a bitwise shift on their operands. They yield a value that is a copy of the left-hand operand with the bits

shifted as directed by the right-hand operand. The right-hand operand must not be negative and must be a value that is strictly less than the number of bits in the result. Otherwise, the operation is undefined. The bits are shifted left (`<<`) or right (`>>`). Bits that are shifted off the end are discarded:

*These illustrations have the low-order bit on the right
These examples assume `char` has 8 bits, and `int` has 32*

// 0233 is an octal literal (§ 2.1.3, p. 38)

`unsigned char bits = 0233; [1 0 0 1 1 0 1 1]`

bits << 8 // bits promoted to `int` and then shifted left by 8 bits

`0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 1 0 0 1 1 0 1 1 | 0 0 0 0 0 0 0 0`

bits << 31 // left shift 31 bits, left-most bits discarded

`1 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0`

bits >> 3 // right shift 3 bits, 3 right-most bits discarded

`0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 1 0 0 1 1`

Figure 2.1: Bitwise Shift Operation

Shift operators have midlevel precedence (lower than the arithmetic operators but higher than the relational, assignment, and conditional operators):

```
1 cout << 42 + 10; // ok: + has higher precedence, so the sum is printed
2 cout << (10 < 42); // ok: parentheses force intended grouping; prints 1
3 cout << 10 < 42; // error: attempt to compare cout to 42!
```

The bitwise NOT operator generates a new value with the bits of its operand inverted:

`unsigned char bits = 0227; [1 0 0 1 0 1 1 1]`

~bits

`[1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 | 0 1 1 0 1 0 0 0]`

Figure 2.2: Bitwise NOT Operation

The AND, OR, and XOR operators generate new values with the bit pattern composed from its two operands:

unsigned char b1 = 0145;	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	0	1	1	0	0	1	0	1	
0	1	1	0	0	1	0	1			
unsigned char b2 = 0257;	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	1	0	1	1	1	1	
1	0	1	0	1	1	1	1			
b1 & b2	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>24 high-order bits all 0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	24 high-order bits all 0	0	0	1	0	0	1	0	1
24 high-order bits all 0	0	0	1	0	0	1	0	1		
b1 b2	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>24 high-order bits all 0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	24 high-order bits all 0	1	1	1	0	1	1	1	1
24 high-order bits all 0	1	1	1	0	1	1	1	1		
b1 ^ b2	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>24 high-order bits all 0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table>	24 high-order bits all 0	1	1	0	0	1	0	1	0
24 high-order bits all 0	1	1	0	0	1	0	1	0		

Figure 2.3: Bitwise AND, OR, and XOR Operation

sizeof Operator

The `sizeof` operator returns the size, in bytes, of an expression or a type name. The operator is right associative. The result of `sizeof` is a constant expression of type `size_t`. The operator takes one of two forms:

```
1 sizeof(type)
2 sizeof(expr)
```

The `sizeof` operator is unusual in that it does not evaluate its operand:

```
1 Sales_data data, *p;
2 sizeof(Sales_data); // size required to hold an object of type Sales_data
3 sizeof data; // size of data's type, i.e., sizeof(Sales_data)
4 sizeof p; // size of a pointer
5 sizeof *p; // size of the type to which p points, i.e., sizeof(Sales_data)
6 sizeof data.revenue; // size of the type of Sales_data's revenue member
7 sizeof Sales_data::revenue; // alternative way to get the size of revenue
```

Dereferencing an invalid pointer as the operand to `sizeof` is safe because the pointer is not actually used, because `sizeof` does not need to dereference the pointer to know what type it will return.

The result of applying `sizeof` depends in part on the type involved:

- `sizeof char` or an expression of type `char` is guaranteed to be 1.
- `sizeof` a reference type returns the size of an object of the referenced type.
- `sizeof` a pointer returns the size needed to hold a pointer.
- `sizeof` a dereferenced pointer returns the size of an object of the type to which the pointer points; the pointer need not be valid.
- `sizeof` an array is the size of the entire array. It is equivalent to taking the `sizeof` of the element type times the number of elements in the array. Note that `sizeof` does not convert the array to a pointer.
- `sizeof` a string or a vector returns only the size of the fixed part of these types; it does not return the size used by the object's elements.

Because `sizeof` returns the size of the entire array, we can determine the number of elements in an array by dividing the array size by the element size.

Comma Operator

The comma operator takes two operands, which it evaluates from left to right. Like the logical AND and logical OR and the conditional operator guarantees the order in which its operands are evaluated. Most common use for the comma operator is in a `for` loop:

```

1 vector<int>::size_type cnt = ivec.size();
2 // assign values from size...1 to the elements in ivec
3 for(vector<int>::size_type ix = 0;
4     ix != ivec.size(); ++ix, --cnt)
5     ivec[ix] = cnt;

```

The left-hand expression is evaluated and its result is discarded. The result of a comma expression is the value of its right-hand expression. The result is an lvalue if the right-hand operand is an lvalue.

Type Conversions

Implicit conversions are carried out automatically without programmer intervention, and are defined to preserve precision, if possible.

- In most expressions, values of integral types smaller than `int` are first promoted to an appropriate larger integral type.
- In conditions, nonbool expressions are converted to `bool`.
- In initializations, the initializer is converted to the type of the variable; in assignments, the right-hand operand is converted to the type of the left-hand.
- In arithmetic and relational expressions with operands of mixed types, the types are converted to a common type.

Conversions also happen during function calls.

Arithmetic conversions:

```

1 bool flag;           char cval;
2 short sval;          unsigned short usval;
3 int ival;            unsigned int uival;
4 long lval;           unsigned long ulval;
5 float fval;          double dval;
6
7 3.14159L + 'a'; // 'a' promoted to int, then that int converted to long double
8 dval + ival; // ival converted to double
9 dval + fval; // fval converted to double
10 ival = dval; // dval converted (by truncation) to int
11 flag = dval; // if dval is 0, then flag is false, otherwise true
12 cval + fval; // cval promoted to int, then that int converted to float

```

```

13 sval + cval; // sval and cval promoted to int
14 cval + lval; // cval converted to long
15 ival + ulval; // ival converted to unsigned long
16 usval + ival; // promotion depends on the size of unsigned short and int
17 uival + lval; // conversion depends on the size of unsigned int and long

```

Array to pointer conversion:

```

1 int ia[10]; // array of ten ints
2 int* ip = ia; // convert ia to a pointer to the first element

```

This conversion is not performed when an array is used with decltype or as the operand of the address-of(&), sizeof, or typeid operators. The conversion is also omitted when we initialize a reference to an array. A similar pointer conversion happens when we use a function type in an expression.

A constant integral value of 0 and the literal nullptr can be converted to any pointer type; a pointer to any nonconst type can be converted to void*, and a pointer to any type can be converted to a const void*

There is an automatic conversion from arithmetic or pointer types to bool. If the pointer or arithmetic value is zero, the conversion yields false; any other yields true:

```

1 char *cp = get_string();
2 if (cp) /* ... */ // trueif the pointer cp is not zero
3 while (*cp) /* ... */ // trueif *cp is not the null character

```

We can convert a pointer to a nonconst type to a pointer to the corresponding const type, and similarly for references. That is, if T is a type, we can convert a pointer or a reference to T into a pointer or a reference to const T:

```

1 int i;
2 const int &j = i; // convert a non const to a reference to const int
3 const int *p = &i; // convert address of a non const to the address of a const
4 int &r = j, *q = p; // error: conversion from const to nonconst not allowed

```

The reverse conversion - removing a low-level const - does not exist.

2.5 December 06, 2023

2.5.1 C++ Statements

An expression becomes an expression statement when it is followed by a semicolon. Expression statements cause the expression to be evaluated and its result discarded:

```

1 ival + 5; // rather useless expression statement
2 cout << ival; // useful expression statement

```

Null statement is a single semicolon, and is legal anywhere a statement is expected:

```

1 ; // null statement
2 ival = v1 + v2;; // ok: second semicolon is a superfluous null statement
3
4 // disaster: extra semicolon: loop body is this null statement
5 while (iter != svec.end()) ; // the while body is the empty statement
6     ++iter; // increment is not part of the loop

```

A compound statement, usually referred to as a block, is a sequence of statements and declarations surrounded by a pair of curly braces. Names introduced inside a block are accessible only in that block and in blocks nested inside that block.

Conditional Statements

An `if` statement conditionally executes another statement based on whether a specified condition is true:

```
1 if (condition)
2     statement
3 else
4     statement2
```

We use a block to enclose multiple statements:

```
1 // if failing grade, no need to check for a plus or minus
2 if (grade < 60)
3     lettergrade = scores[0];
4 else {
5     lettergrade = scores[(grade - 50)/10]; // fetch the letter grade
6     if (grade != 100) // add plus or minus only if not already an A++
7         if (grade % 10 > 7) lettergrade += '+'; // grades ending in 8 or 9 get a +
8         else if (grade % 10 < 3)
9             lettergrade += '-'; // grades ending in 0, 1, or 2 get a -
10 }
```

It is a common mistake to forget the curly braces when multiple statements must be executed as a block.

Dangling `else` is resolved by specifying that each `else` matched with the closest preceding unmatched `if`:

```
1 // WRONG: execution does NOT match indentation; the else goes with the inner if
2 if (grade % 10 >= 3)
3     if (grade % 10 > 7)
4         lettergrade += '+'; // grades ending in 8 or 9 get a +
5 else
6     lettergrade += '-'; // grades ending in 3, 4, 5, 6, or 7 get a minus!
7
8 // add a plus for grades that end in 8 or 9 and a minus for those ending in 0, 1,
9 // or 2
9 if (grade % 10 >= 3) {
10     if (grade % 10 > 7)
11         lettergrade += '+'; // grades ending in 8 or 9 get a +
12 } else // curly braces force the else to go with the outer if
13     lettergrade += '-'; // grades ending in 0, 1, or 2 will get a minus
```

We can make the `else` part of the outer `if` by enclosing the inner `if` in a block.

Iterative Statements

The syntactic form of the `for` statement is:

```
1 // for (initializer; condition; expression)
2 //     statement
```

```

3
4 // process characters in s until we run out of characters or we hit a whitespace
5 for (decltype(s.size()) index = 0;
6     index != s.size() && !isspace(s[index]); ++index)
7     s[index] = toupper(s[index]); // capitalize the current character

```

The order of evaluation of for loop:

1. *init-statement* is executed once at the start of the loop.
2. Next, *condition* is evaluated.
3. If the condition is true, the *for* body executes.
4. Finally, *expression* is evaluated.

init-statement can define several objects in a single declaration statement:

```

1 // remember the size of v and stop when we get to the original last element
2 for (decltype(v.size()) i = 0, sz = v.size(); i != sz; ++i)
3     v.push_back(v[i]);

```

A for header can omit any (or all) of *init-statement*, *condition*, or *expression*, by replacing them with null statements:

```

1 auto beg = v.begin();
2 for (/* null */; beg != v.end() && *beg >= 0; ++beg)
3     ;// no work to do
4
5 // Omitting condition is equivalent to writing true as the condition
6 for (int i = 0; /* no condition */ ; ++i) {
7     // process i; code inside the loop must stop the iteration!
8 }
9
10 // If we omit expression for the for header, either the condition or the body must
11 // do something to advance the iteration
11 vector<int> v;
12 for (int i; cin >> i; /* no expression */)
13     v.push_back(i);

```

The syntactic form of range for statement to iterate through elements of a container or other sequence:

```

1 // for (declaration: expression)
2 //     statement

```

expression must represent a sequence such as braced initializer list, array or object of type such as `vector` or `string` that has `begin` and `end` members that return iterators. *declaration* defines a variable. It must be possible to convert each element of the sequence to the variable's type. The easiest way to make sure the types match is to use the `auto` type specifier.

2.6 December 10, 2023

2.6.1 Learn about SELinux

SELinux Architecture

SELinux consists of four main components: object managers (OM), access vector cache (AVC), security server, and security policy as show below:

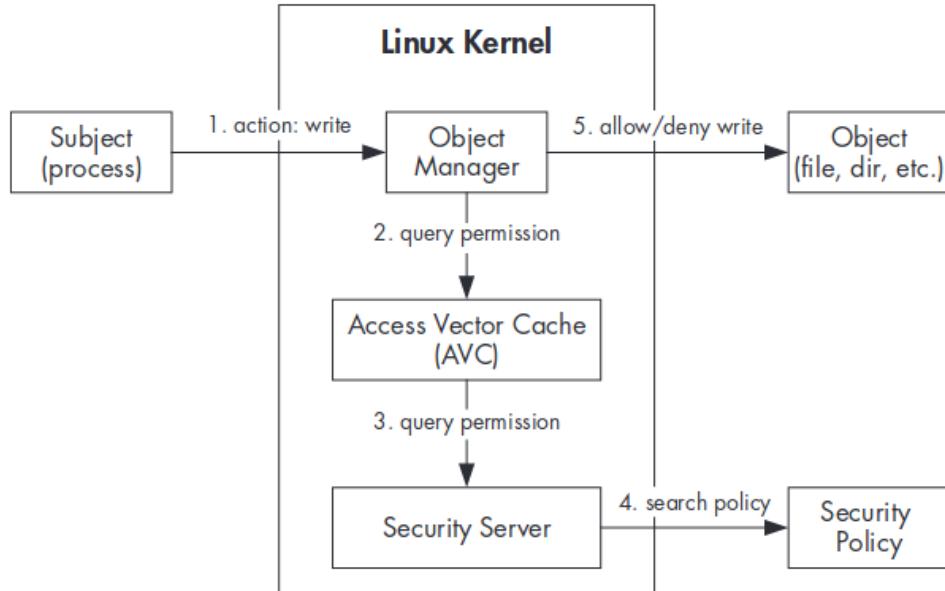


Figure 2.4: SELinux Components

When a subject asks to perform an action on an SELinux object, the associated object manager queries the AVC to see if the attempted action is allowed. If the AVC contains a cached security decision for the request, the AVC returns it to the OM which enforces the decision by allowing or denying the action. If the cache does not contain a matching security decision based on the currently loaded policy and returns it to the AVC, which caches it. The AVC in turn returns it to the OM which ultimately enforces the decision. The security server is part of the kernel, while the policy is loaded from userspace via a series of functions contained in the supporting userspace library.

SELinux Modes

SELinux has 3 modes:

- **Disabled.** No policy is loaded and only the default DAC security is enforced.
- **Permissive.** The policy is loaded and object access is checked, but access denial is only logged - not enforced.

- **Enforcing.** The security policy is both loaded and enforced, with violations logged.

SELinux mode can be checked and changed with the `getenforce` and `setenforce` commands:

```
1 # getenforce
2 Enforcing
3 # setenforce 0
4 # getenforce
5 Permissive
```

The mode set with `setenforce` is not persistent and will be reset to the default mode when the device reboots.

Mandatory Access Control

- **Subjects** are usually running processes that perform actions on objects,
- **Objects** are OS-level resources managed by the kernel (processes can also be objects), and
- **Actions** are carried out only if the security policy allows it.

Both subjects and objects have a set of security attributes (collectively known as the security context) which the OS queries in order to decide whether the requested action should be allowed or not. When SELinux is enabled, subjects cannot bypass or influence policy rules; therefore, the policy is mandatory. The MAC policy is only consulted if the DAC allows access to the resource. If the DAC denies access, the denial is taken as the final security decision.

SELinux support two forms of MAC: *type enforcement (TE)* and *multi-level security (MLS)*. MLS is used to enforce different levels of access to restricted information and is not used in Android. TE implemented in SELinux requires that all subjects and objects have an associated type and SELinux uses this type to enforce the rules of its security policy. A *type* is simply a string that's defined in the policy and associated with objects or subjects. Subject types references processes or groups of processes and are also referred to as *domains*. Types referring to objects usually specify the role an object plays within a policy, such as system file, application data file, and so on. The type (or domain) is an integral part of the security context.

Security Contexts

A *security context* (also referred to as a *security label*, or just *label*) is a string with four fields delimited with colons: username, role, type, and an optional MLS security range.

An SELinux username is typically associated with a group of class of users; for example `user_u` for unprivileged users and `admin_u` for administrators. Users can be associated with one or more domain type. The type is used to group processes in a domain or to specify an object logical type. In Android context, the user is fixed to `u`.

The security range (or level) is used to implement ML and specifies the security levels a subject is allowed to access. In Android context, the security range is fixed to `s0`.

By specifying the option `-Z`, we can see the security context of the processes running:

```
1 # ps -Z
2 u:r:su:s0    root      1847      1834      10842820      3384      sigsuspe+
3 u:r:su:s0    root      1878      1847      10800932      3572      0
```

Subjects inherit the security context of their parent process, or they can change their context via *domain transition* which can be made automatic. For example, all system daemons are started by the `init` process, which has `u:r:init:s0` security context, they would normally inherit this context, but Android's SELinux policy uses automatic domain transitions to set a dedicated domain to each daemon as need.

Similarly the context of files can be revealed using the `-Z` option:

```
1 # ls -Z
2 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 uidgen ->
   toybox
3 -rwxr-xr-x 1 root shell u:object_r:vdc_exec:s0 101920 2023-11-19 00:50 vdc
4 -rwxr-xr-x 1 root shell u:object_r:viewcompiler_exec:s0 277472 2023-11-19 00:50
   viewcompiler
5 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 vmstat ->
   toybox
6 -rwxr-xr-x 1 root shell u:object_r:vold_exec:s0 994368 2023-11-19 00:50 vold
7 -rwxr-xr-x 1 root shell u:object_r:vold_prepare_subdirs_exec:s0 38576 2023-11-19
   00:50 vold_prepare_subdirs
8 -rwxr-xr-x 1 root shell u:object_r:system_file:s0 169 2023-11-19 01:02 vr
9 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 watch ->
   toybox
10 -rwxr-xr-x 1 root shell u:object_r:watchdogd_exec:s0 10760 2023-11-19 00:50
    watchdogd
11 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 wc -> toybox
12 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 which ->
   toybox
13 lrwxr-xr-x 1 root shell u:object_r:system_file:s0 6 2023-11-19 00:50 whoami ->
   toybox
14 -rwxr-xr-x 1 root shell u:object_r:wificond_exec:s0 393248 2023-11-19 00:50
    wificond
```

For objects, the security context is persistent and is usually stored as an extended attribute in the file's metadata. Objects typically inherit the type label of their parent (their directory), and can change to a different label via *type transition*.

Security Policy

Security policies are used by the security server in the kernel to allow or disallow access to kernel objects at runtime. For performance reasons, the policy is typically in binary form generated by compiling a number of policy source files. *Statements* define policy entities such as types, users, and roles. *Rules* allow or deny access to objects (access vector rules); and designate how default users, roles, and types are assigned (default rules). ⁵

The listing below declares `file_type` and `domain` attributes, declares `system_data_file` type and associates it with `file_type` and `data_file_type` attributes, declares `untrusted_app` type and associate it with `domain` attribute:

```

1 attribute file_type;
2 attribute domain;
3
4 type system_data_file, file_type, data_file_type;
5 type untrusted_app, domain;
```

`user` statement declares an SELinux user identifier, associates it with its role(s), and optionally specifies its default security level and the range of security levels that user can access:

```
1 user u roles { r } level s0 range s0 - mls_systemhigh;
```

The `u` user is associated with the `r` role (inside the braces), which in turn is declared using the `role` statement as show below:

```

1 role r;
2 role r types domain;
```

The second statement associates the `r` role with the `domain` attribute, which marks it as a role assigned to processes (domains).

`permissive` statement allows a named domain to run in permissive mode⁶:

```

1 type adbd, domain;
2 permissive adbd;
3 --snip--
```

`class` statement defines an SELinux object class. Object classes and their associated permissions are determined by the respected object manager implementations in Linux kernel, and are static within a policy. Object classes are usually defined in the `security_classes` policy source file:

```

1 --snip-
2 # file-related classes
3 class filesystem
4 class file
5 class dir
6 class fd
7 class lnk_file
```

⁵Type, attribute and permission statements make up the bulk of a security policy.

⁶Most domains in Android's current base policy are permissive.

```

8  class chr_file
9  class blk_file
10 class sock_file
11 class fifo_file
12 --snip--

```

Access vectors are usually defined and associated with object classes in a policy source file called *access_vectors*. Permissions can be either class-specific or inheritable by one or more object classes, in which case they're defined with the `common` keyword. Below is the definition of the set of permissions common to all file objects, and the association of the `dir` class (which represents directories), and a set of directory-specific permissions (`add_name`, `remove_name`, and so on):

```

1  --snip--
2  common file
3  {
4      ioctl
5      read
6      write
7      create
8      getattr
9      setattr
10     lock
11     --snip--
12 }
13 --snip--
14 class dir
15 inherits file
16 {
17     add_name
18     remove_name
19     reparent
20     search
21     rmdir
22     --snip--
23 }
24 --snip--

```

Type Transition Rules

Type enforcement rules and access vector rules typically make the bulk of an SELinux policy. The most commonly used type of enforcement rule is the `type_transition` rule, which specifies when domain and type transitions are allowed:

```

1 # from wpa_supplicant.te
2
3 # wpa - wpa supplicant or equivalent
4 type wpa, domain;
5 permissive wpa;
6 type wpa_exec, exec_type, file_type;
7
8 init_daemon_domain(wpa)
9 unconfined_domain(wpa)
10

```

```

11  # wpa_supplicant daemon uses the type_transition rule to associate the control
      sockets it creates in /data/misc/wifi directory with wpa_socket type
12  type_transition wpa           wifi_data_file: sock_file  wpa_socket;
13  #           source type      target type    class      type of object after
      the transition

```

Domain Transition Rules

Most daemons are associated with a dedicated and use domain transitions to switch their domain when started. This is typically accomplished using the `init_daemon_domain()` macro, which under the hood is implemented using the `type_transition` keyword. The `init_daemon_domain()` macro takes one parameter and is defined in the `te_macros` file using two other macros: `domain_trans()` and `domain_auto_trans()` which are used to allow transition to a new domain and to execute the transition automatically, respectively:

```

1  # Domain transition macros definition int the te_macros file
2
3  # domain_trans(olddomain, type, newdomain)
4  define('domain_trans', '
5  allow $1 $2:file { setattr open read execute };
6  allow $1 $3:process transition;
7  allow $3 $2:file { entrypoint read execute };
8  allow $3 $1:process sigchld;
9  dontaudit $1 $3:process noatsecure;
10 allow $1 $3:process { signinh rlimitinh }; ')
11 # domain_auto_trans(olddomain, type, newdomain)
12 define('domain_auto_trans', '
13 domain_trans($1,$2,$3)
14 type_transition $1 $2:process $3; ')
15 # init_daemon_domain(domain)
16 define('init_daemon_domain', '
17 domain_auto_trans(init, $1_exec, $1)
18 tmpfs_domain($1) ')
19 --snip--

```

The lines beginning with the `allow` keyword are access vector (AV) rules.

Access Vector Rules

AV rules define what privileges processes have at runtime by specifying the set of permissions they have over their target objects:

```

1  # Format of AV rules
2  rule_name source_type target_type : class perm_set;

```

The `rule_name` can be `allow`, `dontallow`, `auditallow`, `neverallow`. `allow` specifies the operations that a subject (process) of the specified source type is allowed to perform on an object of the target type and class specified in the rule. `auditallow` rule is used with `allow` to record audit events when an operation is allowed. `dontaudit` rule is used to suppress the auditing of denial messages when a specified event is known

to be safe. `neverallow` rule says that the declared operation should never be allowed even if an explicit `allow` rule that allows it exists.

To form a rule, `source_type` and `target_type` elements are replaced with one or more previously defined type or attribute identifiers, where `source_type` is the identifier of a subject (process), and `target_type` is the identifier of an object the process is trying to access. The `class` element is replaced with the object class of the target, and `perm_set` specifies the set of permissions that the source process has over the target object. You can specify multiple types, classes, and permissions by enclosing them in braces `{}`). In addition, so rules support use of the wildcard `(*)` and complement(`~`) operators, which allow you to specify that all types should be included or that all types except those explicitly listed should be included, respectively:

```

1 type vold, domain;
2 type vold_exec, exec_type, file_type;
3 init_daemon_domain(vold)
4
5 # allows daemons running in vold domain to mount, unmount, and remount filesystems
   of sdcard_type
6 allow vold sdcard_type:filesystem { mount remount unmount };
7
8 # allows daemons running in vold domain to use the CAP_SYS_PTRACE and CAP_KILL
   Linux capabilities
9 # self means that target domain is same as source (vold in this case)
10 allow vold self:capability { sys_ptrace kill };
11
12 type installd, domain;
13
14 # no audit log will be created if the installd daemon is denied the CAP_SYS_ADMIN
   capability
15 dontaudit installd self:capability sys_admin;
16
17 # forbids all domains but the init domain to load the SELinux policy
18 neverallow { domain -init } kernel:security load_policy;

```

2.7 December 13, 2023

2.7.1 Android Testing

Meeting with Mariano and Biniam

Yan Nai is encountering exceptions in DeepGUI⁷'s final step: MonkeyTest.

DeepGUI has 3 steps:

1. Data Collection of interations and events
2. Training
3. Monkey Test

⁷<https://github.com/Feri73/deep-gui>

2.8 December 19, 2023

2.8.1 SELinux Syntax

SELinux Tutorials from ⁸.

The security context of a process

The security context, together with the run-time user that the process is in, would define what the process is allowed to do.

```
1 ps -ef # -e gives all processes, -f gives more info, -Z gives security context
2
3 ls -ld dir # shows the permissions related to this directory
4 ls -ldZ dir # shows the permissions and also SELinux context
```

- **Domain.** The context of the process that is acting upon something.
- **Type.** The context of the resource on which the process is acting.
- **Class.** The object class of the resource (e.g. *file* or *socket*).
- **Permissions.** The permissions that are allowed given the *domain*, *type* and *class*.

SELinux rule syntax:

```
1 allow <domain> <type>:<class> { <permissions> };
```

Decoding Permission Denial Message

Message:

```
1 type=AVC msg=audit(1363289005.532:184): avc: denied { read } for pid=29199 comm
   = "Trace"
2 name="online" dev="sysfs" ino=30 scontext=staff_u:staff_r:googletalk_plugin_t
3 tcontext=system_u:object_r:sysfs_t tclass=file
```

Log part	Name	Description
type=AVC	Log type	Only in the <i>audit.log</i> file; it informs the user what kind of audit log type this is.

msg=audit(1363289005.532:184) timestamp in seconds since epoch, meaning the number of seconds since January 1st, 1970. You can convert this to a more human readable format using *date -d @* followed by the number, like so: *date -d @1363292159.532*.

⁸<https://wiki.gentoo.org/wiki/SELinux/Tutorials>

Log part	Name	Description
avc:	Log type (again)	
denied	State (if enforced)	What SELinux did, which can be either denied or granted. Note that, if SELinux is in permissive mode, then it will still log as denied even though it was enforced.
{ read }	Permission	The permission that was requested or executed. In this case, it is a read operation. Sometimes the permission contains a set like { read write } but in most cases, it is a single permission request.
for pid=29199	Process PID	The process identifier of the process that took the action.
comm="Trace"	Process CMD	The process command (without arguments, and limited to 15 characters), which helps users identify what the process was in case the process is already gone (a PID is only useful if the process is still running)
name="online"	Target name	The name of the target (in this case, file name). This field depends heavily on the target itself; it can also be path=, capability=, src= and more. But in those cases, its purposes should be clear from the rest of the log.
dev="sysfs"	Device	Device on which the target (in case of a file or file system). In this case, the device is sysfs so we have the hint immediately that this is for something inside /sys. Other valid example are dev=md-0, dev=sda1, or dev=tmpfs.
ino=30	inode number	The inode number of the target file. In this case, since we know it is on the sysfs file system, we can look for this file using: find /sys -xdev -inum 30
scontent=staff u:staff_- r:googletalk_- plugin_t	Source con-	The security context of the process (the domain)
tcontext=syst u:object_- r:sysfs_t	Target con-	The security context of the target resource (in this case the file)
tclass=file	Target class	The class of the target.

Table 2.8: Permission Denied Syntax

2.9 December 22, 2023

2.9.1 SELinux Practical

File Contexts

Objects are mapped to classes⁹ (e.g., a file, a directory, a symbolic link, a socket) and the different kinds of access for each class are represented by permissions. While types and attributes are regularly updated as part of Android SELinux policy, permissions and classes are statically defined and rarely updated as part of a new Linux release.

Attributes

Attributes allow the grouping of access control rules¹⁰. A domain or a type can be assigned an attribute, and access control rules can be defined on attributes. Through attribute, a rule can be valid for all types that are assigned a particular attribute.

2.10 December 24, 2023

2.10.1 Run Daemon

Configure SELinux

Focus on allowing the shell process to execute the

Remove the following from aosp13/device/generic/goldfish/app/wei_daemon/Android.bp:

```
1 init_rc: ["init.weiminn.rc"],
```

Remove the following from AOSP/device/generic/goldfish/sepolicy/x86/weiminn.te:

```
1 type weiminn, domain;
2 #permissive weiminn;
3 type weiminn_exec, vendor_file_type, exec_type, file_type;
4
5 init_daemon_domain(weiminn)
```

Remove the following from AOSP/device/generic/goldfish/app/init.weiminn.rc:

```
1 service weiminn_daemon /vendor/bin/hw/weiminn_daemon
2 class main
3 user system
4 group system
5 oneshot
```

⁹https://android.googlesource.com/platform/system/sepolicy/+/refs/heads/main/private/security_classes

¹⁰https://wiki.gentoo.org/wiki/SELinux/Type_enforcement

Change the file contexts at aosp13/device/generic/goldfish/sepolicy/x86/file_contexts.te:

```
1 /vendor/bin/hw/weiminn_daemon u:object_r:weiminn_bin:s0
```

Making the project returns the following error message:

```
1 [ 1% 9/786] //system/sepolicy:vendor_sepolicy.cil.raw Building cil for
  vendor_sepolicy.cil.raw [common]
2 FAILED: out/soong/.intermediates/system/sepolicy/vendor_sepolicy.cil.raw/
  android_common/vendor_sepolicy.cil.raw
3 out/host/linux-x86/bin/checkpolicy -C -M -c 30 -o out/soong/.intermediates/system/
  sepolicy/vendor_sepolicy.cil.raw/android_common/
4 vendor_sepolicy.cil.raw out/soong/.intermediates/system/sepolicy/vendor_sepolicy.
  conf/android_common/vendor_sepolicy.conf && out/h
5 ost/linux-x86/bin/build_sepolicy filter_out -f out/soong/.intermediates/system/
  sepolicy/reqd_policy_mask_for_vendor.cil/android_co
6 mmon/reqd_policy_mask_for_vendor.cil -t out/soong/.intermediates/system/sepolicy/
  vendor_sepolicy.cil.raw/android_common/vendor_sep
7 olicy.cil.raw # hash of input list: 67
  f02b7bf3aa8ceaff84d183690cbc5581d11fc109b32ddce0ba702ff27c6223
8 device/generic/goldfish/sepolicy/x86/file_contexts.te:2:ERROR 'syntax error' at
  token '/vendor/bin/hw/weiminn_daemon' on line 4329
9 8:
10 /vendor/bin/hw/weiminn_daemon u:object_r:weiminn_bin:s0allow init tmpfs:lnk_file {
  create rename setattr unlink { { getattr open r
11 ead ioctl lock map watch watch_reads } { open append write lock map } } };
12 #line 1 "device/generic/goldfish/sepolicy/x86/file_contexts.te"
13 checkpolicy: error(s) encountered while parsing configuration
14 [ 1% 10/786] //system/sepolicy:sepolicy_neverallows_vendor Neverallow check:
  sepolicy_neverallows_vendor
15 FAILED: out/soong/.intermediates/system/sepolicy/sepolicy_neverallows_vendor/
  policy
16 out/host/linux-x86/bin/checkpolicy -M -c 30 -o out/soong/.intermediates/system/
  sepolicy/sepolicy_neverallows_vendor/policy out/soo
17 ng/.intermediates/system/sepolicy/sepolicy_neverallows_vendor.checkpolicy.conf/
  android_common/sepolicy_neverallows_vendor.checkpol
18 icy.conf # hash of input list: 15871
  fa7f1904c26cf726ea362c19ede7f225bff09f1871b1f7d5657882753a
19 device/generic/goldfish/sepolicy/x86/file_contexts.te:2:ERROR 'syntax error' at
  token '/vendor/bin/hw/weiminn_daemon' on line 7825
20 8:
21 /vendor/bin/hw/weiminn_daemon u:object_r:weiminn_bin:s0allow init tmpfs:lnk_file {
  create rename setattr unlink { { getattr open r
22 ead ioctl lock map watch watch_reads } { open append write lock map } } };
23 #line 1 "device/generic/goldfish/sepolicy/x86/file_contexts.te"
24 checkpolicy: error(s) encountered while parsing configuration
25 16:56:04 ninja failed with: exit status 1
```

Remove the content of aosp13/device/generic/goldfish/sepolicy/x86/file_contexts.te:

```
1 /vendor/bin/hw/weiminn_daemon u:object_r:weiminn_bin:s0
```

Change the service startup script at aosp13/system/core/rootdir/init.rc:

```
1 service weiminn_daemon /vendor/bin/hw/weiminn_daemon
2 class core
3 user shell
4 group shell log readproc
```

```
5 seclabel u:r:shell:s0
```

Now the error message changed to:

```
1 [ 171.154645] init: starting service 'weiminn_daemon'...
2 [ 171.155794] init: cannot execv('/vendor/bin/hw/weiminn_daemon'). See the '
  Debugging init' section of init's README.md for tips: Permission denied
3 [ 171.155852] type=1400 audit(1703410052.492:42): avc: denied { execute } for
  comm="init" name="weiminn_daemon" dev="dm-3" ino=110 scontext=u:r:init:s0
  tcontext=u:object_r:vendor_file:s0 tclass=file permissive=0
4 [ 171.156593] init: Service 'weiminn_daemon' (pid 1868) exited with status 127
5 [ 171.157512] init: Sending signal 9 to service 'weiminn_daemon' (pid 1868)
  process group...
6 [ 171.157946] libprocessgroup: Successfully killed process cgroup uid 2000 pid
  1868 in 0ms
```

inside dmesg, and

```
1 12-24 17:28:50.303      0      0 I init      : starting service 'weiminn_daemon'...
2 12-24 17:28:50.304      0      0 E init      : cannot execv('/vendor/bin/hw/
  weiminn_daemon'). See the 'Debugging init' section of init's README.md for
  tips: Permission denied
3 12-24 17:28:50.305      0      0 I init      : Service 'weiminn_daemon' (pid 1884)
  exited with status 127
4 12-24 17:28:50.306      0      0 I init      : Sending signal 9 to service '
  weiminn_daemon' (pid 1884) process group...
5 12-24 17:28:50.306      0      0 I libprocessgroup: Successfully killed process
  cgroup uid 2000 pid 1884 in 0ms
```

inside adb logcat.

It seems like the daemon is being started by the init process and the binary file inherit its file context vendor_file from its parent folder.

2.11 December 25, 2023

2.11.1 Run Daemon

Configure SELinux

Change the seclabel of the service inside AOSP/device/generic/goldfish/app/init.weiminn.rc to:

```
1 seclabel u:r:shell:s0
from
1 seclabel u:r:weiminn_daemon:s0
```

and the error message changed to

```
1 [61133.720482] init: starting service 'weiminn_daemon'...
2 [61133.721480] init: cannot setexecon('u:r:weiminn_daemon:s0') for weiminn_daemon
  : Invalid argument
3 [61133.722452] init: Service 'weiminn_daemon' (pid 14423) exited with status 6
4 [61133.722745] init: Sending signal 9 to service 'weiminn_daemon' (pid 14423)
  process group...
5 [61133.723107] libprocessgroup: Successfully killed process cgroup uid 0 pid 14423
  in 0ms
```

in dmesg, and

```

1 12-25 10:57:23.010      0      0 I init      : starting service 'weiminn_daemon'...
2 12-25 10:57:23.011      0      0 F init      : cannot setexeccon('u:r:weiminn_daemon:
   s0') for weiminn_daemon: Invalid argument
3 12-25 10:57:23.011      0      0 I init      : Service 'weiminn_daemon' (pid 14441)
   exited with status 6
4 12-25 10:57:23.012      0      0 I init      : Sending signal 9 to service ,
   'weiminn_daemon' (pid 14441) process group...
5 12-25 10:57:23.012      0      0 I libprocessgroup: Successfully killed process
   cgroup uid 0 pid 14441 in 0ms

```

inside adb logcat.

The error seems to be come from the SELinux type `weiminn_daemon` inside the `init.rc`. So, I'm gonna change back to `seclabel u:r:shell:s0`, and inspect the `avc: denied` message, which tells us that the process is `init` type, and the binary is `vendor_file` type.

Add Access Vector rule to allow execute on `vendor_file` type. Alternatively, I put `setenforce 0` inside adb shell, and voila!

Log to logcat every time ART executes a Java method¹¹. But it requires to be NOT on emulator, so scrap that plan.

¹¹<https://stackoverflow.com/questions/33478647/android-app-java-jni-call-hooking-strategies>,
<https://www.youtube.com/watch?v=mFq0vNvUgj8>

Part II

2024

Chapter 3

January

3.1 January 1, 2024

3.1.1 Connect Daemon to ART

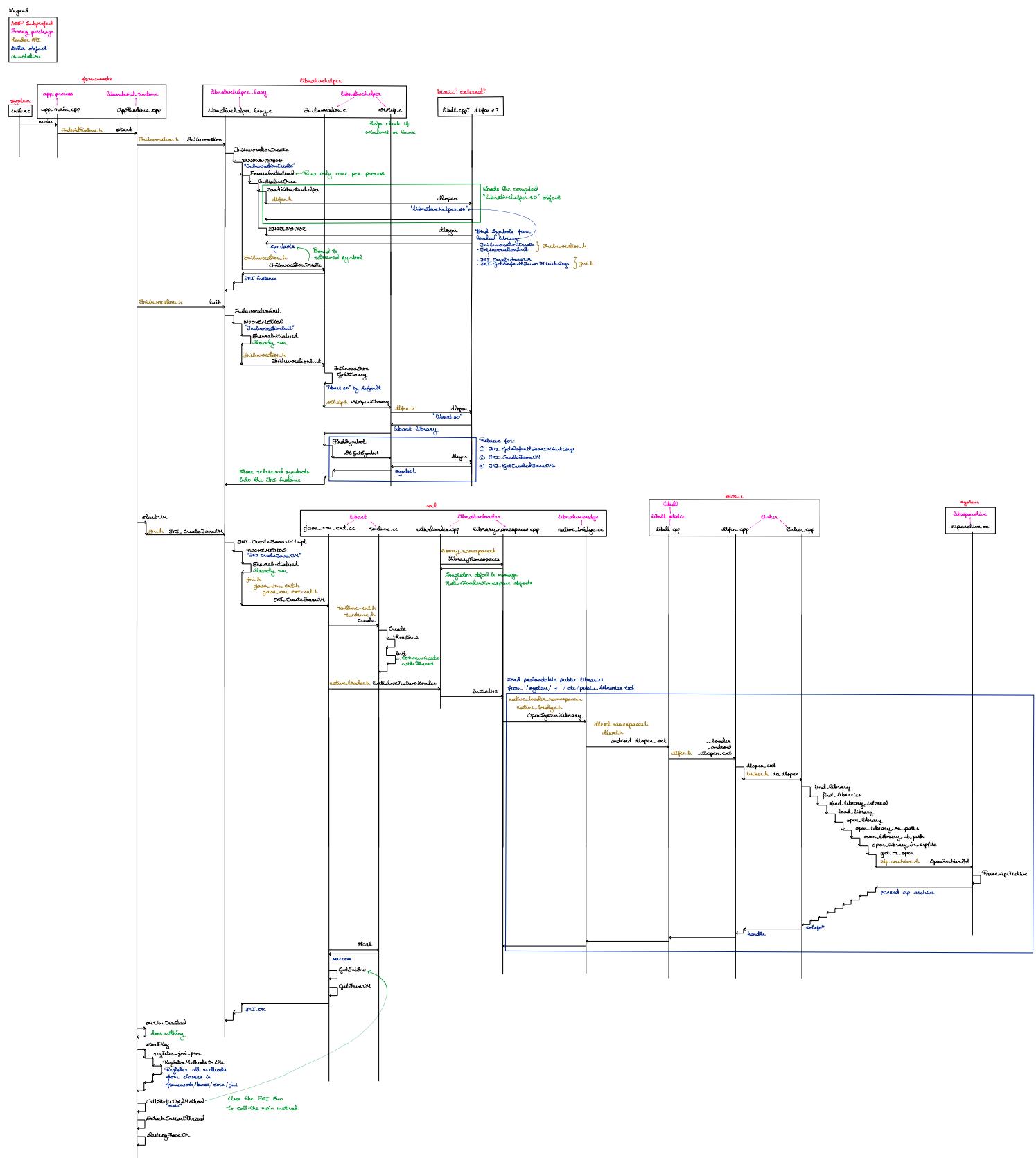
References: Deep dive into ART(Android Runtime) for dynamic binary analysis¹, Android Shared Object loading process ², Android Dynamic Library loading ³

Took me 5 days to properly dissect the ART and Bionic loading process of Android OS.

¹<https://www.youtube.com/watch?v=mFq0vNvUgj8>

²https://blog.csdn.net/qq_37661242/article/details/130448221

³<https://juejin.cn/post/7274991134362124327>



Have to always sync the changes between aosp13/libnativehelper/include_jni/jni.h and aosp13/system/extras/module_ndk_libs/libnativehelper/include_jni/jni.h, so insert the following function definition after the definition of VM Initialization functions:

```

1  /*
2  * VM initialization functions.
3  *
4  * Note these are the only symbols exported for JNI by the VM.
5  */
6  jint JNI_GetDefaultJavaVMInitArgs(void*);
7  jint JNI_CreateJavaVM(JavaVM**, JNIEnv**, void*);
8  jint JNI_GetCreatedJavaVMs(JavaVM**, jsize, jsize*);
9
10 // ADDED BY WEI MINN
11 string WEI_TestFunc();

```

Add k_WEI_TestFunc into MethodIndex enum of aosp13/libnativehelper/libnativehelper_lazy.c.

Add following statement inside the InitializeOnce of aosp13/libnativehelper/libnativehelper_lazy.c after the definition of BIND_SYMBOL:

```

1 // ADDED BY WEI MINN
2 BIND_SYMBOL(WEI_TestFunc);

```

Add following statement at the bottom of aosp13/libnativehelper/libnativehelper_lazy.c:

```

1 // ADDED BY WEI MINN
2 jstring WEI_TestFunc(){
3     return "WEI_TEST_LIBNATIVEHELPER";
4 }

```

Here is the source code of the daemon in aosp13/device/generic/goldfish/app/wei_daemon/weiminn.cpp:

```

1 #include <unistd.h>
2 #include <stdio.h>
3
4 #include <android/log.h>
5 #include <nativehelper/JniInvocation.h>
6 #include <jni.h>
7
8 #define DELAY_SECS 2
9 #define ALOG(msg) __android_log_write(ANDROID_LOG_DEBUG, "WEIMINN_PROJECT", msg)
10
11 int main(int argc, char *argv[]) {
12     ALOG("STARTING_PROJECT");
13
14     int n = 0;
15     while (true) {
16         sleep(DELAY_SECS);
17         n++;
18
19         struct _JNIEnv env;
20         const char *res = env.GetStringUTFChars(WEI_TestFunc(), NULL);
21         ALOG(res);

```

```

22
23     // ALOG("Test");
24 }
25 }
```

But the daemon keeps crashing!

Change the return types from `jstring` to `void*`, and print out `ALOG(WEI_TestFunc())` directly instead of converting from `jstring` to `char` and voila!

Add following statement at the bottom of `art` namespace inside `aosp13/art/runtime/jni/java_vm_ext.cc`:

```

1 // ADDED BY WEI MINN
2 extern "C" const void* WEI_TestFunc() {
3     char const *toReturn = "hello\u00d7from\u00d7runtime";
4     return toReturn;
5 }
```

Add following statement in `JniInvocationInit` function inside `aosp13/libnativehelper/JniInvocation.c`:

```

1 // ORIGINAL
2 DlSymbol JNI_GetDefaultJavaVMInitArgs_ = FindSymbol(library, "JNI_GetDefaultJavaVMInitArgs");
3 if (JNI_GetDefaultJavaVMInitArgs_ == NULL) {
4     return false;
5 }
6
7 DlSymbol JNI_CreateJavaVM_ = FindSymbol(library, "JNI_CreateJavaVM");
8 if (JNI_CreateJavaVM_ == NULL) {
9     return false;
10 }
11
12 DlSymbol JNI_GetCreatedJavaVMs_ = FindSymbol(library, "JNI_GetCreatedJavaVMs");
13 if (JNI_GetCreatedJavaVMs_ == NULL) {
14     return false;
15 }
16
17 // ADDED BY WEI MINN
18 DlSymbol WEI_TestFunc_ = FindSymbol(library, "WEI_TestFunc");
19 if (WEI_TestFunc_ == NULL) {
20     return false;
21 }
22
23 // ORIGINAL
24 instance->jni_provider_library_name = library_name;
25 instance->jni_provider_library = library;
26 instance->JNI_GetDefaultJavaVMInitArgs = (jint (*)(void *)) JNI_GetDefaultJavaVMInitArgs_;
27 instance->JNI_CreateJavaVM = (jint (*)(JavaVM**, JNIEnv**, void*)) JNI_CreateJavaVM_;
28 instance->JNI_GetCreatedJavaVMs = (jint (*)(JavaVM**, jsize, jsize*)) JNI_GetCreatedJavaVMs_;
29
30 // ADDED BY WEI MINN
31 instance->WEI_TestFunc = (char* (*)()) WEI_TestFunc_;
```

Add following statement at the bottom of `JniInvocationImpl` struct in `aosp13/libnativehelper/JniInvocation.c`:

```

1 // Function pointers to methods in JNI provider.
2 jint (*JNI_GetDefaultJavaVMInitArgs)(void* );
3 jint (*JNI_CreateJavaVM)(JavaVM**, JNIEnv**, void* );
4 jint (*JNI_GetCreatedJavaVMs)(JavaVM**, jsize, jsize* );
5
6 // ADDED BY WEI MINN
7 void* (*WEI_TestFunc)();

```

Change all the function return signatures to `const void*`, because of `extern "C"` in `java_vm_ext.cc`.

Trying to call from `libnativehelper_lazy` to art by calling bound method stored inside `INVOKE_METHOD`:

```

1 // ADDED BY WEI MINN
2 const void* WEI_TestFunc(){
3     typedef const void* (*M)();
4     INVOKE_METHOD(WEI_TestFunc, M, library, buffer);
5 }

```

Getting compile time error!

3.2 January 2-3, 2024

3.2.1 Connect ART to Bionic

Need to solve the connection between `libnativehelper_lazy` and `art` first.

Change the parameters of `WEI_TestFunc` inside `libnativehelper_lazy`:

```

1 // ADDED BY WEI MINN
2 const void* WEI_TestFunc(){
3     typedef const void* (*M)();
4     EnsureInitialized();
5     void* method = g_Methods[k_WEI_TestFunc];
6     return ((M) method)();
7 }

```

Got this error message when trying to compile:

```

1 libnativehelper/libnativehelper_lazy.c:287:20: error: array index 16 is past the
      end of the array (which contains 15 elements) [-Werror,-Warray-bounds]
2     void* method = g_Methods[k_WEI_TestFunc];
3

```

Swap the position of `WEI_TestFunc` with `k_MethodCount` inside `MethodIndex` enum of `libnativehelper_lazy.c`:

```

1 // ADDED BY WEI MINN
2 k_WEI_TestFunc,
3
4 // Marker for count of methods
5 k_MethodCount

```

Android Emulator boot at startup screen and this is message from `adb logcat`:

```

1 01-02 13:12:46.793 3799 3799 F zygote : Failed to find symbol 'WEI_TestFunc' in
      libnativehelper.so: undefined symbol: WEI_TestFunc

```

Because I forgot to put the following implementation inside aosp13/libnativehelper/JniInvocation.cpp

```
1 jint WEI_TestFunc(void* res) {
2     ALOG_ALWAYS_FATAL_IF(NULL == g_impl.WEI_TestFunc, "Runtime library not loaded.");
3     return g_impl.WEI_TestFunc(res);
4 }
```

Daemon cannot load libnativehelper.so:

```
1 01-02 13:22:47.648      0      0 I init      : starting service 'weiminn_daemon'...
2 01-02 13:22:48.263  1867  1867 D WEIMINN PROJ: STARTING WEIMINN PROJECT
3 01-02 13:22:50.264  1867  1867 F weiminn_daemon: Failed to load libnativehelper.so
      : dlopen failed: library "libnativehelper.so" not found
```

Tombstone file at adb shell cat /data/tombstones/tombstone_37 > tb.txt:

```

30 00007b83e618d670 00000000282444c7 8d4cf76349fb6348 .D$(....Hc.Ic.L.
31 00007b83e618d680 00000006ba102454 0f050f00000129b8 T$.....).....
32 00007b83e618d690 290ffffc2ad20510 100f00000a02484 ...*....).$.....
33 00007b83e618d6a0 84290ffffc2ab305 b48d480000009024 ...*....).$....H..
34 00007b83e618d6b0 0006bf0000009024 075bcfe8d2310000 $.....1...[.
35 00007b83e618d6c0 894c00000002bf00 00075bb0e8d231f6 .....L..1...[..
36 00007b83e618d6d0 e8c03100000027bf bfc3894800075754 .'.1..TW..H...
37 00007b83e618d6e0 45e8c031000000ba 570fc68949000757 ..)D$...)D$..).$.
38 00007b83e618d6f0 290f102444290fc0 802484290f202444 ..)D$p.)D$‘.)D
39 00007b83e618d700 702444290f000000 44290f602444290f $H.D$(....Hc.Ic.
40 00007b83e618d710 0f402444290f5024 182444c730244429 $P.)D$C.)D$O.D$.
41 00007b83e618d720 20245c89fffffff 24448900075683e8 .....`$ ..V...D$.
42 00007b83e618d730 0000282444c74824 f66349fb63480000 $H.D$(....Hc.Ic.
43 00007b83e618d740 0006ba1024548d4c 0f00000129b80000 L.T$.....).....
44 00007b83e618d750 45e80000007fbf05 ccccccccccccc00075b .....E[.....
45
46 memory near r10 ([stack]):
47 00007fff3fde6d10 000000000000000001 00007b83e618d66b .....k....{..
48 00007fff3fde6d20 c7632a7a8fbc8b50 ffffffffffffdff P...z*c.....
49 00007fff3fde6d30 000000000000000000 00000000fffffff .....'.
50 00007fff3fde6d40 000007d000000bba 0000000000000000 .....'.
51 00007fff3fde6d50 000000000000000000 0000000000000000 .....'.
52 00007fff3fde6d60 000000000000000000 0000000000000000 .....'.
53 00007fff3fde6d70 000000000000000000 0000000000000000 .....'.
54 00007fff3fde6d80 000000000000000000 0000000000000000 .....'.
55 00007fff3fde6d90 000000000000000000 0000000000000000 .....'.
56 00007fff3fde6da0 000000000000000000 0000000000000000 .....'.
57 00007fff3fde6db0 000000000000000000 00007b83e6180998 .....{..
58 00007fff3fde6dc0 00007fff3fde6ed0 00007fff3fde73b8 .n.?....s.?....
59 00007fff3fde6dd0 00007fff3fde6ed0 0000000000000000 .n.?.....
60 00007fff3fde6de0 00007fff3fde73a8 00007b83e67d9a1b .s.?....}...{..
61 00007fff3fde6df0 0000000000000001 00007b83e67da5d3 .....}...{..
62 00007fff3fde6e00 0000003000000020 00007fff3fde72f0 ...0....r.?....
63
64 memory near r12 ([stack]):
65 00007fff3fde7390 000061461a181010 000061461a17f90e ....Fa.....Fa..
66 00007fff3fde73a0 0000000000000001 00007fff3fde827a .....z..?....
67 00007fff3fde73b0 0000000000000000 00007fff3fde8298 .....?.....
68 00007fff3fde73c0 00007fff3fde832d 00007fff3fde8340 -..?....@..?....
69 00007fff3fde73d0 00007fff3fde8355 00007fff3fde8370 U..?....p..?....
70 00007fff3fde73e0 00007fff3fde8383 00007fff3fde839c ...?.....?....
71 00007fff3fde73f0 00007fff3fde83c3 00007fff3fde83ec ...?.....?....
72 00007fff3fde7400 00007fff3fde8419 00007fff3fde8432 ...?....2..?....
73 00007fff3fde7410 00007fff3fde844c 00007fff3fde8467 L..?....g..?....
74 00007fff3fde7420 00007fff3fde8ace 00007fff3fde8cbb ...?.....?....
75 00007fff3fde7430 00007fff3fde8e74 0000000000000000 t..?.....
76 00007fff3fde7440 0000000000000021 00007fff3fdef000 !.....?....
77 00007fff3fde7450 0000000000000033 000000000000006f0 3.....'.
78 00007fff3fde7460 0000000000000010 00000000178afbfd .....'.
79 00007fff3fde7470 0000000000000006 0000000000001000 .....'.
80 00007fff3fde7480 0000000000000011 0000000000000064 .....d.....
81
82 memory near r14 ([stack]):
83 00007fff3fde6d00 00007fff3fde6d28 00007fff3fde73a8 (m.?....s.?....
84 00007fff3fde6d10 0000000000000001 00007b83e618d66b .....k....{..
85 00007fff3fde6d20 c7632a7a8fbc8b50 ffffffffffffdff P...z*c.....
86 00007fff3fde6d30 0000000000000000 00000000fffffff .....'.
87 00007fff3fde6d40 000007d00000bba 0000000000000000 .....'.
88 00007fff3fde6d50 0000000000000000 0000000000000000 .....'.

```

```

89      00007fff3fde6d60 0000000000000000 0000000000000000 ..... .
90      00007fff3fde6d70 0000000000000000 0000000000000000 ..... .
91      00007fff3fde6d80 0000000000000000 0000000000000000 ..... .
92      00007fff3fde6d90 0000000000000000 0000000000000000 ..... .
93      00007fff3fde6da0 0000000000000000 0000000000000000 ..... .
94      00007fff3fde6db0 0000000000000000 00007b83e6180998 ..... {..
95      00007fff3fde6dc0 00007fff3fde6ed0 00007fff3fde73b8 .n.?....s.?...
96      00007fff3fde6dd0 00007fff3fde6ed0 0000000000000000 .n.?.... .
97      00007fff3fde6de0 00007fff3fde73a8 00007b83e67d9a1b .s.?....}..{..
98      00007fff3fde6df0 0000000000000001 00007b83e67da5d3 .....}..{..
99
100     memory near rsp ([stack]): ..... .
101     00007fff3fde6d00 00007fff3fde6d28 00007fff3fde73a8 (m.?....s.?...
102     00007fff3fde6d10 0000000000000001 00007b83e618d66b .....k....{..
103     00007fff3fde6d20 c7632a7a8fbc8b50 ffffffffffffd P...z*c.... .
104     00007fff3fde6d30 0000000000000000 00000000fffffff ..... .
105     00007fff3fde6d40 00007d000000bba 0000000000000000 ..... .
106     00007fff3fde6d50 0000000000000000 0000000000000000 ..... .
107     00007fff3fde6d60 0000000000000000 0000000000000000 ..... .
108     00007fff3fde6d70 0000000000000000 0000000000000000 ..... .
109     00007fff3fde6d80 0000000000000000 0000000000000000 ..... .
110     00007fff3fde6d90 0000000000000000 0000000000000000 ..... .
111     00007fff3fde6da0 0000000000000000 0000000000000000 ..... .
112     00007fff3fde6db0 0000000000000000 00007b83e6180998 ..... {..
113     00007fff3fde6dc0 00007fff3fde6ed0 00007fff3fde73b8 .n.?....s.?...
114     00007fff3fde6dd0 00007fff3fde6ed0 0000000000000000 .n.?.... .
115     00007fff3fde6de0 00007fff3fde73a8 00007b83e67d9a1b .s.?....}..{..
116     00007fff3fde6df0 0000000000000001 00007b83e67da5d3 .....}..{..
117
118     memory near rip (/apex/com.android.runtime/lib64/bionic/libc.so): ..... .
119     00007b83e618d660 45e820245c89ffff 4824244489000757 ...\$ .EW...D$$H
120     00007b83e618d670 00000000282444c7 8d4cf76349fb6348 .D$(....Hc.Ic.L.
121     00007b83e618d680 00000006ba102454 0f050f00000129b8 T$.....).... .
122     00007b83e618d690 290ffffc2ad20510 100f000000a02484 ...*...).$.... .
123     00007b83e618d6a0 84290ffffc2ab305 b48d480000009024 ...*...).$....H..
124     00007b83e618d6b0 0006bf0000009024 075bcfe8d2310000 \$.....1...[. .
125     00007b83e618d6c0 894c0000002bf00 00075bb0e8d231f6 .....L..1...[..
126     00007b83e618d6d0 e8c03100000027bf bfc3894800075754 .'...1..TW..H...
127     00007b83e618d6e0 45e8c031000000ba 570fc68949000757 .....1..EW...I...W
128     00007b83e618d6f0 290f102444290fc0 802484290f202444 ..)D$..)D$...).$..
129     00007b83e618d700 702444290f000000 44290f602444290f .....D$p..)D$`.)D
130     00007b83e618d710 0f402444290f5024 182444c730244429 $P..)D$@..)D$0..D$.
131     00007b83e618d720 20245c89fffffff 24448900075683e8 .....\$ ..V...D$ .
132     00007b83e618d730 0000282444c74824 f66349fb63480000 $H.D$(....Hc.Ic.
133     00007b83e618d740 0006ba1024548d4c 0f00000129b80000 L.T$.....).... .
134     00007b83e618d750 45e80000007fb05 ccccccccc00075b .....E[.... .
135
136     memory map (118 entries): ..... .
137     00006146'1a17e000-00006146'1a17ffff r-- 0 1000 /vendor/bin/hw/
           weiminn_daemon (BuildId: 64dca3fe723864a1e2dd2e698986fb43) (load bias 0
           x1000)
138     00006146'1a17f000-00006146'1a17ffff r-x 0 1000 /vendor/bin/hw/
           weiminn_daemon (BuildId: 64dca3fe723864a1e2dd2e698986fb43) (load bias 0
           x1000)
139     00006146'1a180000-00006146'1a181fff r-- 0 2000 /vendor/bin/hw/
           weiminn_daemon (BuildId: 64dca3fe723864a1e2dd2e698986fb43) (load bias 0
           x1000)
140     00006146'1a182000-00006146'1a182fff rw- 1000 1000 /vendor/bin/hw/
           weiminn_daemon (BuildId: 64dca3fe723864a1e2dd2e698986fb43) (load bias 0
           x1000)

```

```

x1000)
141 00007b81'53015000-00007b81'83a44fff --- 0 30a30000 [anon:cfi shadow]
142 00007b81'83a45000-00007b81'83a45fff r-- 0 1000 [anon:cfi shadow]
143 00007b81'83a46000-00007b81'90c33fff --- 0 d1ee000 [anon:cfi shadow]
144 00007b81'90c34000-00007b81'90c34fff r-- 0 1000 [anon:cfi shadow]
145 00007b81'90c35000-00007b81'9300dfff --- 0 23d9000 [anon:cfi shadow]
146 00007b81'9300e000-00007b81'9300efff r-- 0 1000 [anon:cfi shadow]
147 00007b81'9300f000-00007b81'd3014fff --- 0 40006000 [anon:cfi shadow]
148 00007b81'd3015000-00007b81'd3015fff --- 0 1000
149 00007b81'd3016000-00007b81'd3055fff rw- 0 40000 [anon:scudo:
    primary]
150 00007b81'd3056000-00007b81'e3021fff --- 0 ffcc000
151 00007b81'e3022000-00007b81'e3061fff rw-
    primary] 0 40000 [anon:scudo:
152 00007b81'e3062000-00007b81'f301bfff --- 0 ffba000
153 00007b81'f301c000-00007b81'f305bfff rw-
    primary] 0 40000 [anon:scudo:
154 00007b81'f305c000-00007b82'03019fff --- 0 ffbe000
155 00007b82'0301a000-00007b82'03059fff rw-
    primary] 0 40000 [anon:scudo:
156 00007b82'0305a000-00007b83'e3014fff --- 0 1dffbb000
157 00007b83'e3015000-00007b83'e3063fff r-- 0 4f000 /apex/com.android
    .vndk.v33/lib64/libc++.so (BuildId: fa40b55330a4a2a3eca001f4cae01033) (
    load bias 0x1000)
158 00007b83'e3064000-00007b83'e30c1fff r-x 4e000 5e000 /apex/com.android
    .vndk.v33/lib64/libc++.so (BuildId: fa40b55330a4a2a3eca001f4cae01033) (
    load bias 0x1000)
159 00007b83'e30c2000-00007b83'e30c8fff r-- ab000 7000 /apex/com.android
    .vndk.v33/lib64/libc++.so (BuildId: fa40b55330a4a2a3eca001f4cae01033) (
    load bias 0x1000)
160 00007b83'e30c9000-00007b83'e30c9fff rw- b1000 1000 /apex/com.android
    .vndk.v33/lib64/libc++.so (BuildId: fa40b55330a4a2a3eca001f4cae01033) (
    load bias 0x1000)
161 00007b83'e30ca000-00007b83'e30ccfff rw- 0 3000 [anon:.bss]
162 00007b83'e31cf000-00007b83'e31d1fff r-- 0 3000 /system/lib64/
    libnetd_client.so (BuildId: 0ba64f6d28b890efbf31d3d283147280) (load bias 0
    x1000)
163 00007b83'e31d2000-00007b83'e31d2fff --- 0 1000
164 00007b83'e31d3000-00007b83'e31d6fff r-x 3000 4000 /system/lib64/
    libnetd_client.so (BuildId: 0ba64f6d28b890efbf31d3d283147280) (load bias 0
    x1000)
165 00007b83'e31d7000-00007b83'e31d7fff r-- 6000 1000 /system/lib64/
    libnetd_client.so (BuildId: 0ba64f6d28b890efbf31d3d283147280) (load bias 0
    x1000)
166 00007b83'e31d8000-00007b83'e31d8fff rw- 6000 1000 /system/lib64/
    libnetd_client.so (BuildId: 0ba64f6d28b890efbf31d3d283147280) (load bias 0
    x1000)
167 00007b83'e3200000-00007b83'e5fffffff --- 0 2e00000
168 00007b83'e6057000-00007b83'e6057fff r-- 0 1000 /apex/com.android
    .runtime/lib64/bionic/libdl.so (BuildId: efd82935135bb07f6cf82127470e5900)
    (load bias 0x1000)
169 00007b83'e6058000-00007b83'e6058fff r-x 0 1000 /apex/com.android
    .runtime/lib64/bionic/libdl.so (BuildId: efd82935135bb07f6cf82127470e5900)
    (load bias 0x1000)
170 00007b83'e6059000-00007b83'e6059fff r-- 0 1000 /apex/com.android
    .runtime/lib64/bionic/libdl.so (BuildId: efd82935135bb07f6cf82127470e5900)
    (load bias 0x1000)
171 00007b83'e605a000-00007b83'e605afff --- 0 1000
172 00007b83'e605b000-00007b83'e605bfff r-- 0 1000 [anon:.bss]

```

```

173      00007b83'e6096000-00007b83'e60b4fff r--          0      1f000  /apex/com.android
          .runtime/lib64/bionic/libm.so (BuildId: 9cfa4f2cea749f828172a4bab94dcf95)
          (load bias 0x1000)
174      00007b83'e60b5000-00007b83'e60e5fff r-x      1e000      31000  /apex/com.android
          .runtime/lib64/bionic/libm.so (BuildId: 9cfa4f2cea749f828172a4bab94dcf95)
          (load bias 0x1000)
175      00007b83'e60e6000-00007b83'e60e6fff r--      4e000      1000  /apex/com.android
          .runtime/lib64/bionic/libm.so (BuildId: 9cfa4f2cea749f828172a4bab94dcf95)
          (load bias 0x1000)
176      00007b83'e60e7000-00007b83'e60e7fff rw-      4e000      1000  /apex/com.android
          .runtime/lib64/bionic/libm.so (BuildId: 9cfa4f2cea749f828172a4bab94dcf95)
          (load bias 0x1000)
177      00007b83'e612e000-00007b83'e6171fff r--          0      44000  /apex/com.android
          .runtime/lib64/bionic/libc.so (BuildId: 57def992cb1772e13608c8efcaf893b)
          (load bias 0x1000)
178      00007b83'e6172000-00007b83'e6204fff r-x      43000      93000  /apex/com.android
          .runtime/lib64/bionic/libc.so (BuildId: 57def992cb1772e13608c8efcaf893b)
          (load bias 0x1000)
179      00007b83'e6205000-00007b83'e6209fff r--      d5000      5000  /apex/com.android
          .runtime/lib64/bionic/libc.so (BuildId: 57def992cb1772e13608c8efcaf893b)
          (load bias 0x1000)
180      00007b83'e620a000-00007b83'e620afff rw-      d9000      1000  /apex/com.android
          .runtime/lib64/bionic/libc.so (BuildId: 57def992cb1772e13608c8efcaf893b)
          (load bias 0x1000)
181      00007b83'e620b000-00007b83'e675cff r--          0      552000  [anon:.bss]
182      00007b83'e675d000-00007b83'e675dff r--          0      1000  [anon:.bss]
183      00007b83'e675e000-00007b83'e6765fff rw-          0      8000  [anon:.bss]
184      00007b83'e676f000-00007b83'e67d2fff rw-          0      64000  [anon:
          linker_alloc]
185      00007b83'e67d3000-00007b83'e67d7fff r--          0      5000  /system/lib64/
          liblog.so (BuildId: acde2d0ef136fd6d1eb2a30987f39d55) (load bias 0x1000)
186      00007b83'e67d8000-00007b83'e67e1fff r-x      4000      a000  /system/lib64/
          liblog.so (BuildId: acde2d0ef136fd6d1eb2a30987f39d55) (load bias 0x1000)
187      00007b83'e67e2000-00007b83'e67e2fff r--      d000      1000  /system/lib64/
          liblog.so (BuildId: acde2d0ef136fd6d1eb2a30987f39d55) (load bias 0x1000)
188      00007b83'e67e3000-00007b83'e67e3fff rw-      d000      1000  /system/lib64/
          liblog.so (BuildId: acde2d0ef136fd6d1eb2a30987f39d55) (load bias 0x1000)
189      00007b83'e6800000-00007b83'e95fffff ---          0      2e00000
190      00007b83'e9621000-00007b83'e9640fff r--          0      20000  /dev/
          __properties__/_u:object_r:heapprofd_prop:s0
191      00007b83'e9641000-00007b83'e9660fff r--          0      20000  /dev/
          __properties__/_u:object_r:libc_debug_prop:s0
192      00007b83'e9661000-00007b83'e9669fff r--          0      9000  /apex/com.android
          .vndk.v33/lib64/libcutils.so (BuildId: 43606b529d94720adf378693c75eca35) (
          load bias 0x1000)
193      00007b83'e966a000-00007b83'e9675fff r-x      8000      c000  /apex/com.android
          .vndk.v33/lib64/libcutils.so (BuildId: 43606b529d94720adf378693c75eca35) (
          load bias 0x1000)
194      00007b83'e9676000-00007b83'e9677fff r--      13000      2000  /apex/com.android
          .vndk.v33/lib64/libcutils.so (BuildId: 43606b529d94720adf378693c75eca35) (
          load bias 0x1000)
195      00007b83'e9678000-00007b83'e9678fff rw-      14000      1000  /apex/com.android
          .vndk.v33/lib64/libcutils.so (BuildId: 43606b529d94720adf378693c75eca35) (
          load bias 0x1000)
196      00007b83'e9683000-00007b83'e9696fff r--          0      14000  /apex/com.android
          .vndk.v33/lib64/libbase.so (BuildId: 314908d7976b6d0a21139dcdb1a359d4) (
          load bias 0x1000)
197      00007b83'e9697000-00007b83'e96c2fff r-x      13000      2c000  /apex/com.android
          .vndk.v33/lib64/libbase.so (BuildId: 314908d7976b6d0a21139dcdb1a359d4) (

```

```

        load bias 0x1000)
198 00007b83'e96c3000-00007b83'e96c3fff r--      3e000      1000  /apex/com.android
     .vndk.v33/lib64/libbase.so (BuildId: 314908d7976b6d0a21139dcdb1a359d4) (
     load bias 0x1000)
199 00007b83'e96c4000-00007b83'e96c4fff rw-      3e000      1000  /apex/com.android
     .vndk.v33/lib64/libbase.so (BuildId: 314908d7976b6d0a21139dcdb1a359d4) (
     load bias 0x1000)
200 00007b83'e96c5000-00007b83'e96c5fff rw-      0      1000  [anon:.bss]
201 00007b83'e96e4000-00007b83'e9703fff r--      0      20000  /dev/
     __properties__/_u:object_r:gwp_asan_prop:s0
202 00007b83'e9704000-00007b83'e9752fff r--      0      4f000  /system/lib64/
     libc++.so (BuildId: 7de31e7c740e102016d16ce256f97d3a) (load bias 0x1000)
203 00007b83'e9753000-00007b83'e97b0fff r-x      4e000      5e000  /system/lib64/
     libc++.so (BuildId: 7de31e7c740e102016d16ce256f97d3a) (load bias 0x1000)
204 00007b83'e97b1000-00007b83'e97b7fff r--      ab000      7000  /system/lib64/
     libc++.so (BuildId: 7de31e7c740e102016d16ce256f97d3a) (load bias 0x1000)
205 00007b83'e97b8000-00007b83'e97b8fff rw-      b1000      1000  /system/lib64/
     libc++.so (BuildId: 7de31e7c740e102016d16ce256f97d3a) (load bias 0x1000)
206 00007b83'e97b9000-00007b83'e97bbfff rw-      0      3000  [anon:.bss]
207 00007b83'e97c1000-00007b83'e97e0fff r--      0      20000  /dev/
     __properties__/_u:object_r:debug_prop:s0
208 00007b83'e97e1000-00007b83'e9800fff r--      0      20000  /dev/
     __properties__/_properties_serial
209 00007b83'e9801000-00007b83'e9801fff rw-      0      1000  [anon:
     bionic_alloc_small_objects]
210 00007b83'e981c000-00007b83'e982dfff r--      0      12000  /dev/
     __properties__/_property_info
211 00007b83'e9832000-00007b83'e9851fff r--      0      20000  /dev/
     __properties__/_u:object_r:build_prop:s0
212 00007b83'e9852000-00007b83'e9871fff r--      0      20000  /dev/
     __properties__/_u:object_r:log_tag_prop:s0
213 00007b83'e9872000-00007b83'e9891fff r--      0      20000  /dev/
     __properties__/_u:object_r:vendor_socket_hook_prop:s0
214 00007b83'e9892000-00007b83'e9959fff r--      0      c8000  [anon:
     linker_alloc]
215 00007b83'e995a000-00007b83'e9979fff r--      0      20000  /dev/
     __properties__/_u:object_r:vndk_prop:s0
216 00007b83'e997b000-00007b83'e9982fff rw-      0      8000  [anon:
     bionic_alloc_small_objects]
217 00007b83'e9983000-00007b83'e9983fff rw-      0      1000  [anon:
     bionic_alloc_lob]
218 00007b83'e9984000-00007b83'e9985fff rw-      0      2000  [anon:
     bionic_alloc_small_objects]
219 00007b83'e9986000-00007b83'e9988fff rw-
     property context nodes      0      3000  [anon:System
220 00007b83'e9989000-00007b83'e9989fff ---      0      1000
221 00007b83'e998a000-00007b83'e998cff r--      0      3000  [anon:
     stack_and_tls:main]
222 00007b83'e998d000-00007b83'e998dff ---      0      1000
223 00007b83'e998e000-00007b83'e998ffff rw-
     bionic_alloc_lob      0      2000  [anon:
224 00007b83'e9992000-00007b83'e9996fff rw-
     bionic_alloc_small_objects      0      5000  [anon:
225 00007b83'e9997000-00007b83'e99fafff r--      0      64000  [anon:
     linker_alloc]
226 00007b83'e99fb000-00007b83'e99fbfff rw-
     bionic_alloc_small_objects      0      1000  [anon:
227 00007b83'e99fc000-00007b83'e9a1bfff r--      0      20000  /dev/
     __properties__/_u:object_r:debug_prop:s0

```

```

228      00007b83'e9a1c000-00007b83'e9a3bfff r--      0      20000  /dev/
229          __properties__/_u:object_r:build_prop:s0
230      00007b83'e9a3c000-00007b83'e9a3cff  ---      0      1000
231      00007b83'e9a3d000-00007b83'e9a44fff rw-      0      8000
232      00007b83'e9a45000-00007b83'e9a45fff ---      0      1000
233      00007b83'e9a46000-00007b83'e9a65fff r--      0      20000  /dev/
234          __properties__/_properties_serial
235          property_context_nodes]
236      00007b83'e9a66000-00007b83'e9a68fff rw-      0      3000  [anon: System
237          _properties__/_property_info
238          linker_alloc]
239      00007b83'e9a7b000-00007b83'e9adefff r--      0      64000  [anon:
240          bionic_alloc_small_objects]
241      00007b83'e9ae1000-00007b83'e9ae1fff r--      0      1000  [anon: atexit
242          handlers]
243      00007b83'e9ae2000-00007b83'e9ae2fff ---      0      1000
244      00007b83'e9ae3000-00007b83'e9aeafff rw-      0      8000  [anon: thread
245          signal_stack]
246      00007b83'e9aeb000-00007b83'e9aebfff rw-      0      1000  [anon: arc4random
247          data]
248      00007b83'e9aec000-00007b83'e9aecfff rw-      0      1000  [anon: abort
249          message]
250      00007b83'e9aed000-00007b83'e9aedfff r--      0      1000  [anon: atexit
251          handlers]
252      00007b83'e9aee000-00007b83'e9aeeee r--      0      1000  [anon: arc4random
253          data]
254      00007b83'e9aef000-00007b83'e9b32fff r--      0      44000  /apex/com.android
255          .runtime/bin/linker64 (BuildId: 88b9a0b1efb561bd3d51b1972ee9b024) (load
256          bias 0x1000)
257      00007b83'e9b33000-00007b83'e9c27fff r-x      43000  f5000  /apex/com.android
258          .runtime/bin/linker64 (BuildId: 88b9a0b1efb561bd3d51b1972ee9b024) (load
259          bias 0x1000)
260      00007b83'e9c28000-00007b83'e9c2ffff r--      137000  8000  /apex/com.android
261          .runtime/bin/linker64 (BuildId: 88b9a0b1efb561bd3d51b1972ee9b024) (load
262          bias 0x1000)
263      00007b83'e9c30000-00007b83'e9c31fff rw-      13e000  2000  /apex/com.android
264          .runtime/bin/linker64 (BuildId: 88b9a0b1efb561bd3d51b1972ee9b024) (load
265          bias 0x1000)
266      00007b83'e9c32000-00007b83'e9c3afff rw-      0      9000  [anon:.bss]
267      00007b83'e9c3b000-00007b83'e9c3bfff r--      0      1000  [anon:.bss]
268      00007b83'e9c3c000-00007b83'e9c3dfff rw-      0      2000  [anon:.bss]
269      00007fff'3fdc8000-00007fff'3fde8fff rw-      0      21000  [stack]
270      00007fff'3fdeb000-00007fff'3fdeefff r--      0      4000  [vvar]
271      00007fff'3fdef000-00007fff'3fdeffff r-x      0      1000  [vdso]
272      ffffffff'ff600000-ffffffffff'ff600fff --x      0      1000  [vsyscall]
273      ----- tail end of log main
274 01-02 13:41:59.096 3002 3002 D WEIMINN PROJ: STARTING WEIMINN PROJECT
275 01-02 13:42:01.096 3002 3002 F weiminn_daemon: Failed to load libnativehelper.so
276          : dlopen failed: library "libnativehelper.so" not found
277
278 279  open files:
280      fd 0: /dev/null (unowned)
281      fd 1: /dev/null (unowned)
282      fd 2: /dev/null (unowned)
283      fd 3: socket:[57035] (unowned)
284      fd 4: /dev/pmsg0 (unowned)
285 286  ----- log main

```

```
266 01-02 13:41:59.096 3002 3002 D WEIMINN PROJ: STARTING WEIMINN PROJECT
267 01-02 13:42:01.096 3002 3002 F weiminn_daemon: Failed to load libnativehelper.so
      : dlopen failed: library "libnativehelper.so" not found
```

I think it's because we are calling into `libnativehelper.so` without loading it first using `JniInvocation`. So we need to figure out the way to initialize `JNI inovation` and store the instance.

Test out calling just `libnativehelper.c` instead of all the way into `art modules`.

3.3 January 4-7, 2024

3.3.1 Load Native Shared Libraries to Daemon

Permanently set SELinux to permissive

Go to `aosp/system/core/init/selinux.cpp` and change `StatusFromProperty()` and `IsEnforcing()` to return `PERMISSIVE` hardcodedly:

```
1 EnforcingStatus StatusFromProperty() {
2     return SELINUX_PERMISSIVE; //in early stage, the function returns permissive
3     status
4     EnforcingStatus status = SELINUX_PERMISSIVE;
5     ImportKernelCmdline([&](const std::string& key, const std::string& value) {
6         if (key == "androidboot.selinux" && value == "permissive") {
7             status = SELINUX_PERMISSIVE;
8         }
9     });
10    if (status == SELINUX_ENFORCING) {
11        status = SELINUX_PERMISSIVE;
12    }
13    return SELINUX_PERMISSIVE;
14 }
15
16 bool IsEnforcing() {
17     return false; //selinux returns false under any enforcing circumstances.
18     if (ALLOW_PERMISSIVE_SELINUX) {
19         return StatusFromProperty() == SELINUX_PERMISSIVE;
20     }
21     return true;
22 }
```

Configure Daemon VNDK

According to Android VNDK documentation⁴, the library that want to use VNDK or LLNDK cannot be related to ART.

⁴<https://source.android.com/docs/core/architecture/vndk?hl=en#framework-shared-libraries> and https://source.android.com/static/docs/core/architecture/images/vndk_design_android_o.pdf

Tried to assign SELinux context to the daemon⁵, but gave up cos there might be further work to do regarding shared library access.

Changing libnativehelper to llndk or vndk in it's android.bp also doesn't help.

Making libnativehelper apex_available is even worse because it will mess up your code and your Android Emulator won't start up even after fixing the code.

So, it's time to give up and just get on with the Schema sharing that Zichen told us to do.

Connect to Runtime

Copy and paste the customized files into the AOSP folder to overwrite.

Tried to set it up for emulator build but a bit too difficult, so I'll just use the given Pixel 7 Pro. So, download driver for Pixel 7 Pro:

```
1 curl -O https://dl.google.com/dl/android/aosp/google_devices-cheetah-td1a
      .220804.009.a2-724bfafcf.tgz
2 tar -xvzf google_devices-cheetah-td1a.220804.009.a2-724bfafcf.tgz
3 ./extract-google_devices-cheetah.sh
```

Reclone the AOSP repo because the latest branch of Android 13 is not compatible with Minima. Don't delete the whole aosp13 source folder, instead go inside and delete subfolders apart from .repo and . Afterwards, clone the different version:

```
1 git ls-remote -h https://android.googlesource.com/platform/manifest.git
2 mkdir aosp && cd aosp
3 repo init -u https://android.googlesource.com/platform/manifest -b android-13.0.0
      _r11 # For Android 13
4 repo sync -c -j$(nproc --all) # Sync the current branch using all available
      processing units in the system
5 source build/envsetup.sh
6 lunch aosp_cheetah-userdebug
7 make -j31
8 adb reboot bootloader # Make sure your device's bootloader is unlocked before
      proceeding.
9 fastboot -w flashall
```

Changed the value of LOG_NDEBUG to 0 that is defined on top of aosp/frameworks/base/core/jni/AndroidRuntime.cpp.

Connect Runtime to Bionic

Need to figure out art's native_bridge.cc talks to bionic's libdl.cpp.

Some of the headers are prebuilt inside aosp/prebuilt. Just ignore them cos you can find the real headers inside bionic, and art.

Okay, time to give up on adding my own function. Instead just try to sneak pointer through the flow. Changes in the arguments will be a challenge.

⁵<https://discuss.96boards.org/t/run-executable-from-init-service-with-root-permissions/8257/2>

Added extra function because overloading gives a lot of error. But I'm still having error because I was extending a shared system library:

```

1 lib_name: "libdl"
2 arch: "arm64"
3 functions_added {
4     return_type: "void *"
5     function_name: "android_dlopen_ext2"
6     source_file: "bionic/libc/include/android/dlext.h"
7     parameters {
8         referenced_type: "const char *"
9         default_arg: false
10        is_this_ptr: false
11    }
12    parameters {
13        referenced_type: "int"
14        default_arg: false
15        is_this_ptr: false
16    }
17    parameters {
18        referenced_type: "const android_dlextinfo *"
19        default_arg: false
20        is_this_ptr: false
21    }
22    parameters {
23        referenced_type: "void *"
24        default_arg: false
25        is_this_ptr: false
26    }
27    linker_set_key: "android_dlopen_ext2"
28    access: public_access
29 }
30 added_elf_functions {
31     name: "android_dlopen_ext2"
32 }
33 compatibility_status: EXTENSION

```

Added extra function because overloading gives a lot of error. Added the function name to bionic/libdl/libdl.map.txt, and compile. But I'm still having error because I was extending a shared system library:

```

1 lib_name: "libdl"
2 arch: "arm64"
3 functions_added {
4     return_type: "void *"
5     function_name: "android_dlopen_ext2"
6     source_file: "bionic/libc/include/android/dlext.h"
7     parameters {
8         referenced_type: "const char *"
9         default_arg: false
10        is_this_ptr: false
11    }
12    parameters {
13        referenced_type: "int"
14        default_arg: false
15        is_this_ptr: false
16    }
17    parameters {

```

```

18     referenced_type: "const android_dlextinfo *"
19     default_arg: false
20     is_this_ptr: false
21   }
22   parameters {
23     referenced_type: "void *"
24     default_arg: false
25     is_this_ptr: false
26   }
27   linker_set_key: "android_dlopen_ext2"
28   access: public_access
29 }
30 added_elf_functions {
31   name: "android_dlopen_ext2"
32 }
33 compatibility_status: EXTENSION

```

Go to `prebuilts/abi-dumps/ndk/33/64/arm64/source-based/libdl.so.1sdump` and in the following functions array, we can only see the entry for the old `android_dlopen_ext`:

```

1 {
2   "function_name" : "android_dlopen_ext",
3   "linker_set_key" : "android_dlopen_ext",
4   "parameters" :
5   [
6     {
7       "referenced_type" : "_ZTIPKc"
8     },
9     {
10       "referenced_type" : "_ZTIi"
11     },
12     {
13       "referenced_type" : "_ZTIPK17android_dlextinfo"
14     }
15   ],
16   "return_type" : "_ZTIPv",
17   "source_file" : "bionic/libc/include/android/dlext.h"
18 },

```

and same goes for `elf_functions` array:

```

1 {
2   "binding" : "weak",
3   "name" : "android_dlopen_ext"
4 },

```

The problem turns out to be the the builder checking for ABI compatibility, so run this command to dump your changes to the ABI database:

```

1 python3 development/vndk/tools/header-checker/utils/create_reference_dumps.py -l
    libdl

```

Given up on overloading the function. Now, modifying `android_extinfo` whose instances are passed from `OpenSystemLibrary` in `art/libnativebridge/native_bridge.cc` to `android_dlopen_ext` in `bionic/libdl/libdl.cpp`. Have to run the ABI dump after modifying the struct of `android_extinfo`.

The `android_dlopen_ext` object is passed from `OpenSystemLibrary` all the way to `do_dlopen` in `linker.cpp`. But for some reason, it is not possible to access the members of the `dlext` object as the object itself is `null`, so the code below is unusable:

```
1 LD_LOG(kLogErrors, "[weiminn] linker.cpp RECEIVED FROM NATIVE BRIDGE!!!: %s",
    extinfo->weiminn_msg);
```

Using the code will cause Android stuck in Recovery mode in infinite loop, even though the project successfully compiles.

Successfully called a custom function inside `linker_phdr.cpp` from `dlfcn.cpp`. So, we are skipping `linker.cpp` which is not required by Zicheng anyway.

3.4 January 11-15, 2024

3.4.1 Dronlomaly Tool Demo Paper

Setup Development Environment on Ubuntu 23

Download `ffmpeg` from Ubuntu official source site⁶. Installing on Ubuntu 23, need to patch `ffmpeg` according to some forum⁷ where we have to fix some assembly code like:

```
1 /*
2  * simple math operations
3  * Copyright (c) 2006 Michael Niedermayer <michaelni@gmx.at> et al
4  *
5  * This file is part of FFmpeg.
6  *
7  * FFmpeg is free software; you can redistribute it and/or
8  * modify it under the terms of the GNU Lesser General Public
9  * License as published by the Free Software Foundation; either
10 * version 2.1 of the License, or (at your option) any later version.
11 *
12 * FFmpeg is distributed in the hope that it will be useful,
13 * but WITHOUT ANY WARRANTY; without even the implied warranty of
14 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
15 * Lesser General Public License for more details.
16 *
17 * You should have received a copy of the GNU Lesser General Public
18 * License along with FFmpeg; if not, write to the Free Software
19 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
20 */
21
22 #ifndef AVCODEC_X86_MATHOPS_H
23 #define AVCODEC_X86_MATHOPS_H
24
25 #include "config.h"
26
27 #include "libavutil/common.h"
28 #include "libavutil/x86/asm.h"
```

⁶<https://launchpad.net/ubuntu/+source/ffmpeg/7:3.4.11-0ubuntu0.1>

⁷bbs.archlinux.org/viewtopic.php?id=289424

```

29
30 #if HAVE_INLINE_ASM
31
32 #if ARCH_X86_32
33
34 #define MULL MULL
35 static av_always_inline av_const int MULL(int a, int b, unsigned shift)
36 {
37     int rt, dummy;
38     if (__builtin_constant_p(shift))
39     __asm__ (
40         "imull%3\uuuuuuuuuuuuuuu\u\n\t"
41         "shrdl%4, %edx, %eax\u\n\t"
42         : "=a"(rt), "=d"(dummy)
43         // :"a"(a), "rm"(b), "ci"((uint8_t)shift)
44         :"a"(a), "rm"(b), "i"(shift & 0x1F)
45     );
46     else
47     __asm__ (
48         "imull%3\uuuuuuuuuuuuu\u\n\t"
49         "shrdl%4, %edx, %eax\u\n\t"
50         : "=a"(rt), "=d"(dummy)
51         :"a"(a), "rm"(b), "c"((uint8_t)shift)
52     );
53
54     return rt;
55 }
56
57 #define MULH MULH
58 static av_always_inline av_const int MULH(int a, int b)
59 {
60     int rt, dummy;
61     __asm__ (
62         "imull%3"
63         : "=d"(rt), "=a"(dummy)
64         :"a"(a), "rm"(b)
65     );
66     return rt;
67 }
68
69 #define MUL64 MUL64
70 static av_always_inline av_const int64_t MUL64(int a, int b)
71 {
72     int64_t rt;
73     __asm__ (
74         "imull%2"
75         : "=A"(rt)
76         :"a"(a), "rm"(b)
77     );
78     return rt;
79 }
80
81 #endif /* ARCH_X86_32 */
82
83 #if HAVE_I686
84 /* median of 3 */
85 #define mid_pred mid_pred
86 static inline av_const int mid_pred(int a, int b, int c)
87 {

```

```

88     int i=b;
89     __asm__ (
90         "cmp%2,%1\n\t"
91         "cmovg%1,%0\n\t"
92         "cmovg%2,%1\n\t"
93         "cmp%3,%1\n\t"
94         "cmovl%3,%1\n\t"
95         "cmp%1,%0\n\t"
96         "cmovg%1,%0\n\t"
97         : "+&r"(i), "+&r"(a)
98         : "r"(b), "r"(c)
99     );
100    return i;
101 }
102
103 #if HAVE_6REGS
104 #define COPY3_IF_LT(x, y, a, b, c, d) \
105 __asm__ volatile(\
106     "cmpl%0,%3\n\t" \
107     "cmovl%3,%0\n\t" \
108     "cmovl%4,%1\n\t" \
109     "cmovl%5,%2\n\t" \
110     : "+&r" (x), "+&r" (a), "+r" (c) \
111     : "r" (y), "r" (b), "r" (d) \
112 );
113 #endif /* HAVE_6REGS */
114
115 #endif /* HAVE_I686 */
116
117 #define MASK_ABS(mask, level) \
118     __asm__ ("cdq\n\t" \
119             "xord%1,%0\n\t" \
120             "subl%1,%0\n\t" \
121             : "+a"(level), "=d"(mask))
122
123 // avoid +32 for shift optimization (gcc should do that ...)
124 #define NEG_SSR32 NEG_SSR32
125 static inline int32_t NEG_SSR32( int32_t a, int8_t s){
126     if (_builtin_constant_p(s))
127         __asm__ ("sarl%1,%0\n\t" \
128             : "+r" (a)
129             // : "ic" ((uint8_t)(-s))
130             : "i" (-s & 0x1F)
131     );
132     else
133         __asm__ ("sarl%1,%0\n\t" \
134             : "+r" (a)
135             : "c" ((uint8_t)(-s))
136             // : "i" (-s & 0x1F)
137     );
138     return a;
139 }
140
141 #define NEG_USR32 NEG_USR32
142 static inline uint32_t NEG_USR32(uint32_t a, int8_t s){
143     if (_builtin_constant_p(s))
144         __asm__ ("shrl%1,%0\n\t" \
145             : "+r" (a)
146             // : "ic" ((uint8_t)(-s))

```

```

147      : "i" (-s & 0x1F)
148  );
149  else
150      __asm__ ("sarl%1,%0\n\t"
151      : "+r" (a)
152      : "c" ((uint8_t)(-s))
153      // : "i" (-s & 0x1F)
154  );
155  return a;
156 }
157
158 #endif /* HAVE_INLINE_ASM */
159 #endif /* AVCODEC_X86_MATHOPS_H */

```

Afterwards, you should be able to install:

```

1 ./configure
2 make
3 sudo make install

```

Running Dronlomaly

Need to fix some compile time error that is leftover from previous year, and also the resultant of migration of Ubuntu 23.

Add `#include <cstdlib>` to `osdk-core/modules/src/filemgr/impl/downloadbufferqueue.cpp`.

Now, need to install the RPC and Boost library:

```
1 sudo apt install libxmlrpc-c++8-dev libboost-all-dev
```

Got to `build/sample/platform/linux/telemetry/CMakeFiles/djiosdk-telemetry-sample.dir/link.txt` and add `-lm -lz -lswresample` to the end of the first line, and copy `UserConfig.txt` at the root of the project into `build/bin/`, and run `./djiosdk-telemetry-sample`:

```
1 cd ~/Documents/x1_backup/DronLomaly/1_TelemetrySubscriber/ && rm -rf build &&
   mkdir build && cd build && cmake .. && make && cp ../../UserConfig.txt bin
```

Since we have to modify the CMakeFiles, we are not supposed to `rm` the whole directory anymore, so, we'll just run `make` to update the binaries after making changes in the code.

I fucked up and installed Python 3.7 from source and the versions are mixed up right now. Successfully reversed it using:

```

1 rm -f /usr/local/bin/python3.7
2 sudo rm -f /usr/local/bin/python3.7
3 sudo rm -f /usr/local/bin/pip3.7
4 sudo rm -f /usr/local/bin/pydoc
5 sudo rm -rf /usr/local/include/python3.7m/
6 sudo rm -f /usr/local/lib/libpython3.7m.a
7 sudo rm -rf /usr/local/lib/python3.7
8 sudo rm -rf /usr/local/lib/pkgconfig/
9 sudo rm -f /usr/local/bin/idle
10 sudo rm -f /usr/local/bin/easy_install-2.7

```

Packages need to run:

```
1 pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org
   /whl/cpu
2 pip3 install matplotlib pandas
3 pip3 install pyqt5
4 sudo apt install python3-tk
```

It works! The key is to NOT install anything qt-related and install python3-tk instead.

Validate the figures with experiment

3.5 January 16-20, 2024

3.5.1 Explore Migration of Zicheng's Works to AOSP14

Clone AOSP14

Tried cloning AOSP 14 into new directory:

```
1 mkdir aosp14 && cd aosp14
2 repo init -u https://android.googlesource.com/platform/manifest -b android-14.0.0.
   r_21
```

But faced this error:

```
1 manifests:
2 fatal: couldn't find remote ref refs/heads/android-14.0.0.r_21
3 manifests: sleeping 4.0 seconds before retrying
4 fatal: cannot obtain manifest https://android.googlesource.com/platform/manifest
5 =====
6 Repo command failed: UpdateManifestError
7     Unable to sync manifest default.xml
```

Checking the available branches with:

```
1 git ls-remote -h https://android.googlesource.com/platform/manifest.git
```

And the result is:

```
1 ...
2 06e1764f504727281253313fbf6c227cebd65e4c refs/heads/android-13.0.0_r79
3 2f7b75202188e6deee159b2dea97719a51f63c2f refs/heads/android-13.0.0_r8
4 4ca496b8ea717f861e193233cd3917d9fbbb5a7 refs/heads/android-13.0.0_r80
5 259db423b147ea6f75b25a87a47190a17b1fda10 refs/heads/android-13.0.0_r81
6 ed487956fff8d6b616d99ace142203d6b89aa877 refs/heads/android-13.0.0_r82
7 f5d5c1675fabbca4cebc0d1b618ff6f7b6f14025 refs/heads/android-13.0.0_r83
8 0bac787fd054b65de0e755dd8f32b1f07cc0b485 refs/heads/android-13.0.0_r9
9 e3f9241982f55ef4cec97df0c55e9b87d37a5656 refs/heads/android-14.0.0_r1
10 32f134303b62e8815287a61ad285cc0171fb3091 refs/heads/android-14.0.0_r10
11 29fddb193223ac701779eb45a926b21e9e6c9585 refs/heads/android-14.0.0_r11
12 1ea2801d54e826ea03e3a6c33e09c2c36da4df41 refs/heads/android-14.0.0_r12
13 ab70a6753ba6dfa223eec03762de67a3e001111f refs/heads/android-14.0.0_r13
14 d774d22bad0a0dcc3494566b52f48b40afb2c0dd refs/heads/android-14.0.0_r14
15 323da1dc43d2c9ace9f97ddf5744fdbcb5c0d4e7 refs/heads/android-14.0.0_r15
16 e43bd2ae656e593e68015f922b7f5b1af335bf0c refs/heads/android-14.0.0_r16
17 d091bf89b3c2e459dc46fe4238695205990228a8 refs/heads/android-14.0.0_r17
18 dcd243f609d59f6d919660898949775d0976ed0b refs/heads/android-14.0.0_r18
```

```

19 bd6b6c3427247e3cdf23c4daafdd4e050488aadb refs/heads/android-14.0.0_r19
20 cae010b8c39e45a9fc675fce989a284c5e39ebcd refs/heads/android-14.0.0_r2
21 d25293c4ced217d523f7773a3426fee6e8c5160c refs/heads/android-14.0.0_r20
22 ce85fafc1db1bafbe98e326a9ac1fde211f0495d refs/heads/android-14.0.0_r21
23 ...

```

The branch of our current AOSP 13 according to `.repo/manifests/default.xml` is:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2   <manifest>
3
4     <remote  name="aosp"
5           fetch=".."
6           review="https://android-review.googlesource.com/" />
7     <default revision="refs/tags/android-13.0.0_r11"
8           remote="aosp"
9           sync-j="4" />

```

Instead of switching branch to a totally different android version, try switching to another branch of 13:

```

1 cd aosp13
2 repo init -u https://android.googlesource.com/platform/manifest -b android-13.0.0_r83

```

Also fail! STUPID ERROR! should `_r83`, NOT `.r_83`! NOW FIXED! Cloning into new directory now.

3.6 January 21-31, 2024

3.6.1 Understand Comprehensive Android Invocation Flow

System Server

Startup process⁸

ServiceManager vs SystemServiceManager⁹

Looper¹⁰ is at the end of System Server to pop messages from message queue and invoke the corresponding methods to handle it.”

System UI

WindowManagerService¹¹

Loading of System UI¹²

⁸<https://blog.csdn.net/q1210249579/article/details/128793307>

⁹<https://blog.csdn.net/b1480521874/article/details/79422909>

¹⁰<https://stackoverflow.com/questions/35931899/why-main-threads-looper-loop-doesnt-block-ui-thread>

¹¹<https://blog.csdn.net/CallmeZhe/article/details/113989896>

¹²<https://stackoverflow.com/questions/37465026/when-systemui-loads-in-android-boot>

Activity Manager

The Activity manager is started by Android's Zygote's System Services

Android application launch process¹³

Role of ActivityManager¹⁴

Role¹⁵ of ActivityThread¹⁶

Role¹⁷ of Instrumentation¹⁸

Activity Record¹⁹

Tasks²⁰

Starting Activity²¹

Start Activity to Zygote Flow²²

ART Invocation

Method Execute Flow²³

Dex2Oat²⁴

¹³<https://multi-core-dump.blogspot.com/2010/04/android-application-launch.html>, <https://multi-core-dump.blogspot.com/2010/04/android-application-launch-part-2.html>, and <https://blog.csdn.net/u012481172/article/details/49658633>

¹⁴<https://developer.android.com/reference/android/app/ActivityManager#nested-classes>

¹⁵<https://blog.csdn.net/hzwailll/article/details/85339714>

¹⁶<https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/app/ActivityThread.java>

¹⁷<https://stackoverflow.com/questions/29140889/when-is-instrumentation-involved-in-app-start>

¹⁸<https://developer.android.com/reference/android/app/Instrumentation>

¹⁹<https://stackoverflow.com/questions/18127984/what-is-activity-record-object-in-android>

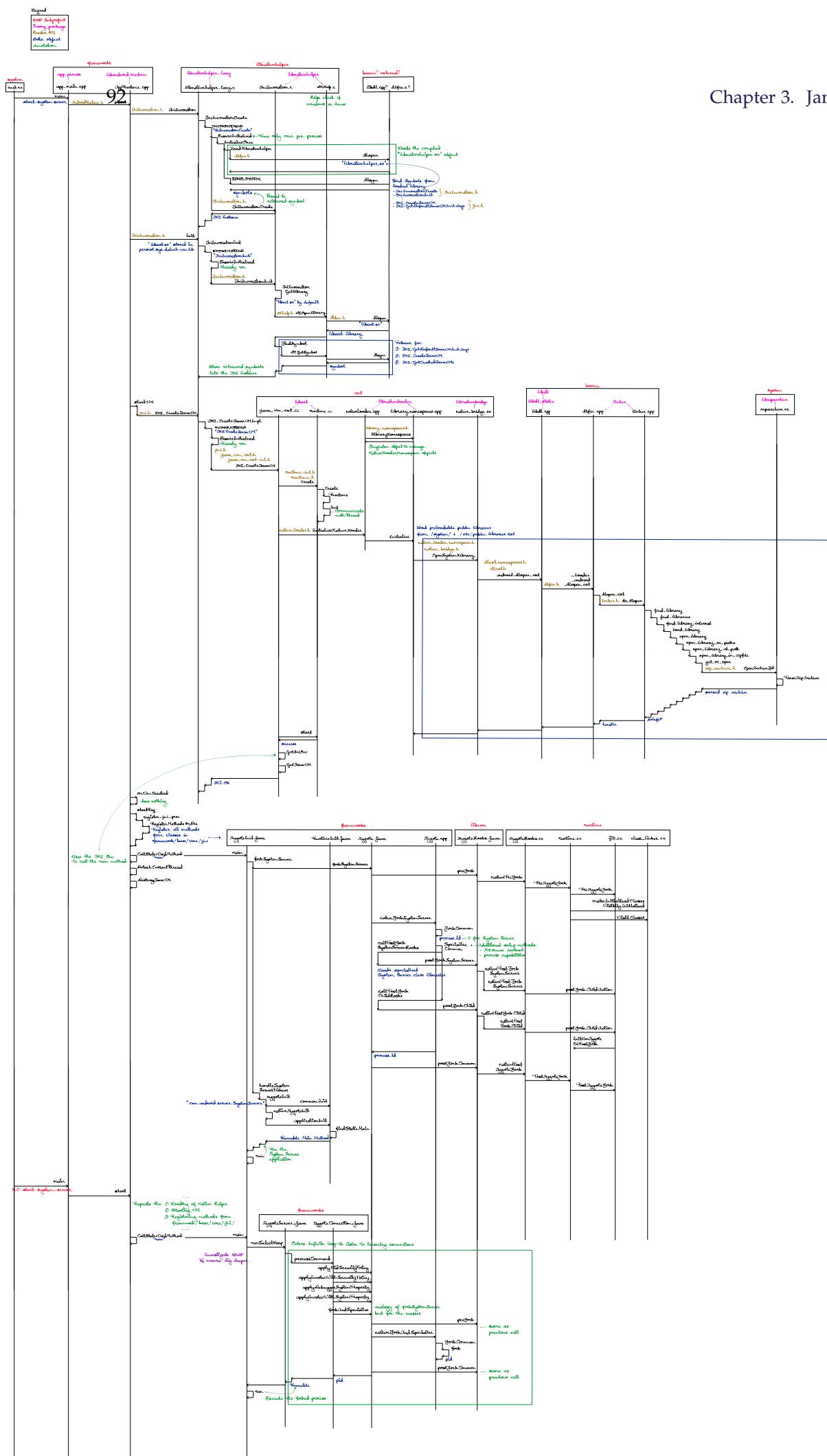
²⁰<https://www.youtube.com/watch?v=MvIlVsXxXmY>

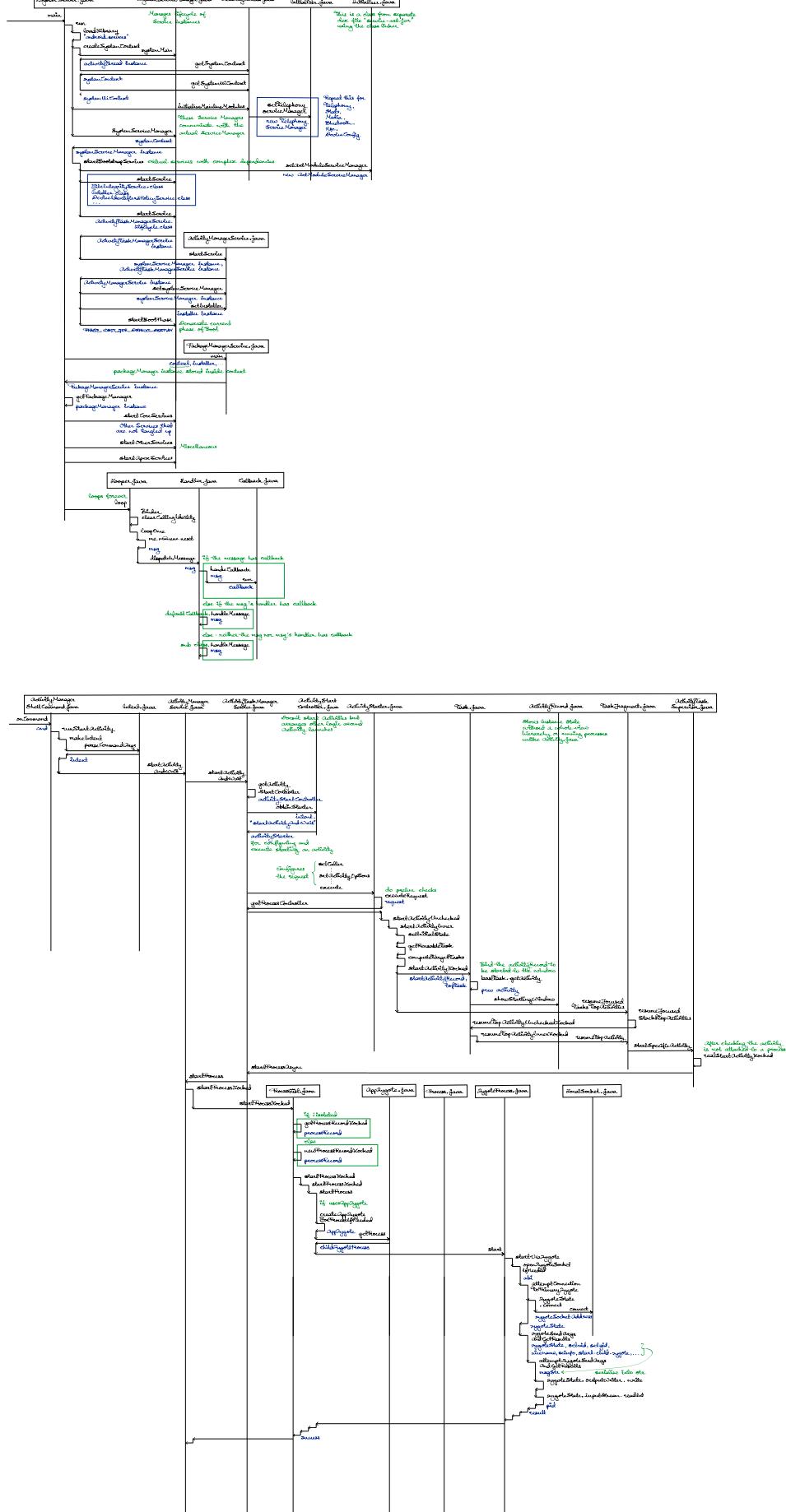
²¹<https://duanqz.github.io/2016-07-29-Activity-LaunchProcess-Part1>

²² [https://github.com/Marukohe/blog-notes/issues/4]

²³https://huanle19891345.github.io/en/android/art/jni/java_jni%E6%96%B9%E6%B3%95%E8%B0%83%E7%94%A8%E5%8E%9F%E7%90%86/

²⁴<https://huanle19891345.github.io/en/android/art/1%E7%B1%BB%E7%BC%96%E8%AF%91/dex2oat/>





Chapter 4

Febuary

4.1 Febuary 11-12, 2024

4.1.1 Understand Papers on Android Framework Level

Characterizing and Detecting Configuration Compatibility Issues in Android Apps (ASE 2021)

Identify compatibility issues from many XML configuration files, caused by inconsistent processing of attributes across different Android APIs. It's also hard to identify code changes causing such inconsistencies. Path-sensitive analysis is accurate but expensive. Android Developers and API Differences Reports can miss many configuration changes that manifest runtime inconsistencies.

Empirical Study. Data collected from bug-related code revisions from well-maintained open-source Android apps.

- **RQ1: Issue types and root causes.** Unavailable configuration APIs, Inconsistent configuration APIs, Inconsistent Android internal XML configuration files, Inconsistent attribute dependencies, Inconsistent attribute usages, and Inconsistent attribute default values.
- **RQ2: Issue symptoms.** 45.4% crash, 44.9% inconsistent look-and-feel.

ConfDroid.

- **Configuration Constraint Model.** To deal with inconsistencies in APIs that handle configuration, constraint tuple $\{\mathcal{A}, \mathcal{F}, \mathcal{X}\}$ is defined where \mathcal{A} stands for attribute, \mathcal{X} stands for XML tag \mathcal{A} is located in, and \mathcal{F} stands for data format assignable to \mathcal{A} .
- **Build Trimmed ICFG.** The problematic code changes reside in the class that invokes configurations APIs. The call graphs and each method's CFGs are combined to give Trimmed ICFG.

- Extract Android Configuration Constraints.
- Generate Detection Rules.

Evaluation.

- RQ3: Effectiveness of detection rule extraction.

Baselines: Lint¹, Orplocator², cDep³

- RQ4: Usefulness.

Compatibility Issue Detection for Android Apps Based on Path-Sensitive Semantic Analysis (ICSE 2023)

Compatibility Issues due to deprecated APIs. Difficult to extract constraints for different check patterns across different classes and methods, and semantic analysis of constraints in each API usage path considering API lifetime.

PSDroid.

1. **Call Graph Generation**. Extract call graph from the API using Soot, and FlowDroid.
2. **Path Extraction**. Distinguish the APIs used by the main packages or TPLs in the app.
3. **API Usage Check Pattern Analysis**.
4. **Path-sensitive Analysis**.
5. **API Lifetime Modeling**.
6. **API Compatibility Issue Detection**.

Experiments.

1. **RQ1: Detecting different types of API compatibility issues**.
2. **RQ2: Detecting API compatibility issues effectively**.
3. **RQ3: Outperforming existing tools on detecting API compatibility issues**.
4. **RQ4: Feedbacks on issue reports**.

Limitations. Inherits limitations from static analysis in terms of reflection handling, native code, multi-threading, and unresolved types. False positives on dead codes. Cannot solve complex constraints. Questionable accuracy of API lifetime modelling. Bias of manual analysis.

¹<https://developer.android.com/studio/write/lint>

²https://zhendong2050.github.io/res/ORPLocator_ISSRE_2016.pdf

³<https://www.cs.cornell.edu/~legunsen/pubs/ChenETAL20CDep.pdf>

Natidroid: Cross-Language Android Permission Specification (FSE 2022)

Invocation chain: Initialize `CameraManager` and call its `openCamera()` inside the SDK which calls `CameraService.cpp` in the native library. This is not detected by SOTA (Axplorer⁴, Arcade⁵, Pscout⁶) as they only analyze the Java framework, not native codes.

NatiDroid. Addresses the cross-language permission mapping problem.

1. **Pre-processing.** Prepare the intermediate artifacts from the AOSP codebase using Soot and Clang to allow the analysis of Android Framework.
2. **Entry-points Identification.**
 - AIDL-based communication as the IPC mechanism for Java classess to communicate with native libraries in a client-server form.
 - JNI-based communication as a dynamic linker for Java and C++ classes to call each other.
3. **Cross-language CFG Generation.** Leverage forward analysis on the native side from each identified entry-point. If it includes security checkpoint, continue to do backward analysis to build Java-side CFG. Afterwards, connect the 2 CFGs.
4. **Protection mapping extraction.** Perform DFS from Java node to find security checks in the native layer.

Evaluation.

1. **Protection Mapping in Android Native Libraries.**
2. **Applications of Protection Mappings.**

Permission Over-privilege Detection.
Component Hijacking Detection.

Discussion and Limitations.

1. **Android Versions.**
2. **Custom ROMs.**
3. **Static Analysis.**

⁴https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_backes-android.pdf

⁵<https://www.cs.purdue.edu/homes/taog/papers/CCS18.pdf>

⁶<https://security.csl.toronto.edu/papers/PScout-CCS2012-web.pdf>

4.2 Febuary 13, 2024

4.2.1 Conduct Experiments on AOSP 14

Mine Permission Mappings for AOSP 14

Set up Java and compile the Java code:

```

1 sudo apt install openjdk-8-jdk-headless    # version 8u382-ga-1ubuntu1
2 rm -rf out && mkdir out && javac src/*.java src/javaLink/*.java src/cppLink/*.java
      src/util/*.java -cp "./libs/*" -d out && java -cp "./out/../libs/*"
      findJavaLink
3 sudo update-alternatives --config java

```

Gets error message:

```

1 Exception in thread "main" java.lang.RuntimeException: java.util.concurrent.
   ExecutionException: java.lang.Exception: Error: The path '/Library/Java/
   JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home/jre/lib/rt.jar' does not
   exist.
2   at soot.SourceLocator.getClassSourceType(SourceLocator.java:312)
3   at soot.SourceLocator.lookupInClassPath(SourceLocator.java:615)
4   at soot.asm.AsmClassProvider.find(AsmClassProvider.java:39)
5   at soot.SourceLocator.getClassSource(SourceLocator.java:187)
6   at soot.SootResolver.bringToHierarchyUnchecked(SootResolver.java:231)
7   at soot.SootResolver.bringToHierarchy(SootResolver.java:221)
8   at soot.SootResolver.bringToSignatures(SootResolver.java:292)
9   at soot.SootResolver.bringToBodies(SootResolver.java:332)
10  at soot.SootResolver.processResolveWorklist(SootResolver.java:171)
11  at soot.SootResolver.resolveClass(SootResolver.java:141)
12  at soot.Scene.forceResolve(Scene.java:2095)
13  at javaLink.JavaLink.run(JavaLink.java:78)
14  at findJavaLink.main(findJavaLink.java:14)
15 Caused by: java.util.concurrent.ExecutionException: java.lang.Exception: Error:
   The path '/Library/Java/JavaVirtualMachines/jdk1.8.0_231.jdk/Contents/Home/jre
   /lib/rt.jar' does not exist.
16  at com.google.common.util.concurrent.AbstractFuture.getDoneValue(
   AbstractFuture.java:552)
17  at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java
   :513)
18  at com.google.common.util.concurrent.AbstractFuture$TrustedFuture.get(
   AbstractFuture.java:90)
19  at com.google.common.util.concurrent.Uninterruptibles.getUninterruptibly(
   Uninterruptibles.java:197)
20  at com.google.common.cache.LocalCache$Segment.getAndRecordStats(LocalCache.
   java:2229)
21  at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2195)
22  at com.google.common.cache.LocalCache$Segment.lockedGetOrLoad(LocalCache.java
   :2153)
23  at com.google.common.cache.LocalCache$Segment.get(LocalCache.java:2043)
24  at com.google.common.cache.LocalCache.get(LocalCache.java:3851)
25  at com.google.common.cache.LocalCache.getOrLoad(LocalCache.java:3875)
26  at com.google.common.cache.LocalCache$LocalLoadingCache.get(LocalCache.java
   :4800)
27  at soot.SourceLocator.getClassSourceType(SourceLocator.java:310)
28  ... 12 more
29 Caused by: java.lang.Exception: Error: The path '/Library/Java/JavaVirtualMachines
   /jdk1.8.0_231.jdk/Contents/Home/jre/lib/rt.jar' does not exist.

```

```

30      at soot.SourceLocator$1.load(SourceLocator.java:73)
31      at soot.SourceLocator$1.load(SourceLocator.java:68)
32      at com.google.common.cache.LocalCache$LoadingValueReference.loadFuture(
33          LocalCache.java:3445)
33      at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2194)
34  ... 18 more

```

So, find the available Java libraries in our machine using `sudo find / -iname rt.jar`:

```

1 /home/weiminn/Documents/aosp13/prebuilts/jdk/jdk8/darwin-x86/jre/lib/rt.jar
2 /home/weiminn/Documents/aosp13/prebuilts/jdk/jdk8/linux-x86/jre/lib/rt.jar
3 /home/weiminn/Documents/aosp14/prebuilts/jdk/jdk8/darwin-x86/jre/lib/rt.jar
4 /home/weiminn/Documents/aosp14/prebuilts/jdk/jdk8/linux-x86/jre/lib/rt.jar
5 find: '/proc/1246576': No such file or directory
6 /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/rt.jar
7 find: '/run/user/1000/gvfs': Permission denied
8 find: '/run/user/1000/doc': Permission denied

```

Change the following contents of `Config.java` to:

```

1 //      static String MAC_jreDir = "/Library/Java/JavaVirtualMachines/jdk1.8.0_231.
2 //          jdk/Contents/Home/jre/lib/rt.jar";
3 //      static String MAC_jceDir = "/Library/Java/JavaVirtualMachines/jdk1.8.0_231.
4 //          jdk/Contents/Home/jre/lib/jce.jar";
5
6 static String MAC_jreDir = "/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/rt.jar";
7 static String MAC_jceDir = "/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jce.jar";
8
9 //      static String MAC_ANDROID_JAR = "/android-hidden-api-master/android-26/
10 //          android.jar";
11 //      static String MAC_SERVICES_JAR = "/android-hidden-api-master/android-26/
12 //          services.jar";
13
14 static String MAC_ANDROID_JAR = "/home/weiminn/Android/Sdk/platforms/android-34/
15 //          android.jar";
15 static String MAC_SERVICES_JAR = "/home/weiminn/Documents/aosp14/out/target/
16 //          product/emulator-x86_64/system/framework/services.jar";

```

Error message:

```

1 Exception in thread "main" java.lang.RuntimeException: java.util.concurrent.
2     ExecutionException: java.lang.Exception: Error: The path '/home/weiminn/
3     Documents/NatiDroid/jar8.1/services.devicepolicy_intermediates.jar' does not
4     exist.
5     at soot.SourceLocator.getClassSourceType(SourceLocator.java:312)
6     at soot.SourceLocator.lookupInClassPath(SourceLocator.java:615)
7     at soot.asm.AsmClassProvider.find(AsmClassProvider.java:39)
8     at soot.SourceLocator.getClassSource(SourceLocator.java:187)
9     at soot.SootResolver.bringToHierarchyUnchecked(SootResolver.java:231)
10    at soot.SootResolver.bringToHierarchy(SootResolver.java:221)
11    at soot.SootResolver.bringToSignatures(SootResolver.java:292)
12    at soot.SootResolver.bringToBodies(SootResolver.java:332)
13    at soot.SootResolver.processResolveWorklist(SootResolver.java:171)
14    at soot.SootResolver.resolveClass(SootResolver.java:141)
15    at soot.Scene.forceResolve(Scene.java:2095)
16    at javaLink.JavaLink.run(JavaLink.java:78)
17    at findJavaLink.main(findJavaLink.java:14)
18
19 Caused by: java.util.concurrent.ExecutionException: java.lang.Exception: Error:
20     The path '/home/weiminn/Documents/NatiDroid/jar8.1/services.

```

```

16     devicepolicy_intermediates.jar' does not exist.
17     at com.google.common.util.concurrent.AbstractFuture.getDoneValue(
18         AbstractFuture.java:552)
18     at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java
19         :513)
19     at com.google.common.util.concurrent.AbstractFuture$TrustedFuture.get(
20         AbstractFuture.java:90)
20     at com.google.common.util.concurrent.Uninterruptibles.getUninterruptibly(
21         Uninterruptibles.java:197)
20     at com.google.common.cache.LocalCache$Segment.getAndRecordStats(LocalCache.
21         java:2229)
21     at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2195)
22     at com.google.common.cache.LocalCache$Segment.lockedGetOrLoad(LocalCache.java
23         :2153)
23     at com.google.common.cache.LocalCache$Segment.get(LocalCache.java:2043)
24     at com.google.common.cache.LocalCache.get(LocalCache.java:3851)
25     at com.google.common.cache.LocalCache.getOrLoad(LocalCache.java:3875)
26     at com.google.common.cache.LocalCache$LocalLoadingCache.get(LocalCache.java
27         :4800)
27     at soot.SourceLocator.getClassSourceType(SourceLocator.java:310)
28     ... 12 more
29 Caused by: java.lang.Exception: Error: The path '/home/weiminn/Documents/NatiDroid
30     /jar8.1/services/devicepolicy_intermediates.jar' does not exist.
30     at soot.SourceLocator$1.load(SourceLocator.java:73)
31     at soot.SourceLocator$1.load(SourceLocator.java:68)
32     at com.google.common.cache.LocalCache$LoadingValueReference.loadFuture(
33         LocalCache.java:3445)
33     at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2194)
34     ... 18 more

```

Modify /home/weiminn/Documents/NatiDroid/py/collect_jar.py:

```

1 version = '7.0'
2 project_path = '/android-7.0.0_r33/out/target/common/obj/JAVA_LIBRARIES'
3
4 version = '14.0'
5 project_path = '/home/weiminn/Documents/aosp14/out/target/common/obj/
    JAVA_LIBRARIES',

```

4.3 Febuary 14-15, 2024

4.3.1 Figure out NatiDroid for AOSP14

Solved Java part of Natidroid

```

1 git clone https://github.com/Sable/soot.git
2 git checkout develop
3 sudo apt install maven
4 sudo update-alternatives --config java
5 mvn clean compile assembly:single
6 rm -rf out && mkdir out && javac src/*.java src/javaLink/*.java src/cppLink/*.java
    src/util/*.java -cp "./libs/*" -d out && java -cp "./out:/soot/target/
    sootclasses-trunk-jar-with-dependencies.jar" findJavaLink

```

Workspace:

```
1 # terminal 1
2 rm -rf out && mkdir out && javac src/*.java src/javaLink/*.java src/cppLink/*.java
   src/util/*.java -cp "./soot/target/sootclasses-trunk-jar-with-dependencies.
   jar:./libs/fastjson-1.2.73.jar" -d out && java -cp "./out:/soot/target/
   sootclasses-trunk-jar-with-dependencies.jar" findJavaLink
3 # terminal 2
4 clear && find ./ \( -iname '*.java' -a ! -iname '*test*' \) | xargs grep --color -
   in 'internalTransform.*;'
5 # terminal 3
6 mvn clean compile assembly:single
```


Chapter 5

March

5.1 March 05-13, 2024

5.1.1 Migrate Zicheng's code

Code differencing

file	13 to Debloated 13	13 to 14
art_dex_file_loader.cc	No change	Removed MemMapContainer. Replace DexFileLoader::OpenCommon now protected with ArtDexFileLoader::Open, and remove its other overloads.
dex_file_loader.cc	No change	Remove DexFileLoader::GetMultiDexChecksums, DexFileLoader::OpenAll, DexFileLoader::OpenOneDexFileFromZip, DexFileLoader::OpenAllDexFilesFromZip. Modified DexFileLoader::Open and added its overloads, DexFileLoader::OpenCommon and added its overloads. Added DexFileLoader::InitAndReadMagic, DexFileLoader::OpenWithDataSection, DexFileLoader::OpenFromZipEntry
native_loader.cpp	Trivial loggings	No change

file	13 to Debloated 13	13 to 14
native_loader_namespace.cpp	Trivial loggings	No change
odrefresh.cc	Change from args.emplace_back("--compiler-filter=speed"); to args.emplace_back("--compiler-filter=verify");	Tons of change
quick_trampoline_entrypoints.cc	No change	Lots of change
interpreter_common.cc	Trivial Logging	Modified ShouldStayInSwitchInterpreter, MoveToExceptionHandler, DoCallCommon, ArtInterpreterToCompiledCodeBridge, InvokeBootstrapMethod, DoCall. Added UnlockHeldMonitors
interpreter_switch.cc	Trivial JNI headers imported	Modified ShouldStayInSwitchInterpreter, MoveToExceptionHandler, DoCallCommon, ArtInterpreterToCompiledCodeBridge, InvokeBootstrapMethod, DoCall. Added UnlockHeldMonitors

file	13 to Debloated 13	13 to 14
interpreter_ switch_ impl-inl.h	Trivial JNI headers imported	Modified template settings for CheckTransactionAbort, CheckForceReturn, HandlePendingException, HandleReturn, HandleGet, HandlePut, HandleInvoke, RETURN_OBJECT, MONITOR_ENTER, MONITOR_EXIT, NEW_ARRAY, FILLED_NEW_ARRAY, FILLED_NEW_ARRAY_RANGE, ExecuteSwitchImplCpp. Modified Preamble, CONST_CLASS, CHECK_CAST, INSTANCE_OF, NEW_INSTANCE, THROW, CHECK_CAST. Added DoAssignabilityChecks
interpreter_ switch_ impl.h	Trivial logging headers imported	Modified template settings for ExecuteSwitchImplCpp, ExecuteSwitchImpl.
interpreter. cc	No changes	Modified ExecuteSwitch, Execute, EnterInterpreterFromInvoke, EnterInterpreterFromDeoptimize, EnterInterpreterFromEntryPoint.
jit.cc, jit_code_ cache.cc	No changes	Changes

file	13 to Debloated 13	13 to 14
java_vm_ext.cc	Modified FindSymbol to check if MINIMA symbol	Modified JavaVMExt::JavaVMExt, JavaVMExt::Create, JavaVMExt::AddGlobalRef, JavaVMExt::AddWeakGlobalRef, JavaVMExt::DeleteGlobalRef, JavaVMExt::DeleteWeakGlobalRef, JavaVMExt::DisallowNewWeakGlobals, JavaVMExt::AllowNewWeakGlobals, JavaVMExt::DecodeGlobal, JavaVMExt::GetIndirectRefKind, JavaVMExt::DecodeWeakGlobalDuringShutdown, JavaVMExt::LoadNativeLibrary, JavaVMExt::FindCodeForNativeMethod, JavaVMExt::GetLibrarySearchPath. Added JavaVMExt::Initialize, JavaVMExt::DecodeWeakGlobalAsStrong. Remove JavaVMExt::SweepJniWeakGlobals

Table 5.1: Differences

Code differencer

```

1  import os
2  import shutil
3  import difflib
4
5  def html_prep(files, title, workspace, out):
6      html_output = """
7          <!DOCTYPE html>
8          <html>
9              <head>
10                  <style>
11                      table, th, td {{
12                          border: 1px solid black;
13                      }}
14                  </style>
15              </head>
16              <body>
17                  <h2>{}</h2>
18              </body>
19
20              <table>
21                  <thead>
22                      <tr>
23                          <th>{}</th>

```

```
24         </tr>
25     </thead>
26     <tbody>
27     """ .format(title, out)
28
29     for file in files:
30         html_output += """
31         <tr>
32             <td><a href="{}_diff.html">{}</a></td>
33         </tr>""".format(file, file[len(workspace)+out:])
34
35     # Close HTML tags
36     html_output += "</table></body></html>"
37     return html_output
38
39 def compare_files(file1, file2, workspace):
40
41     """
42         Compares two C++ source files using the difflib module.
43
44     Args:
45         file1: Path to the first C++ source file.
46         file2: Path to the second C++ source file.
47
48     Returns:
49         A string containing the differences between the two files,
50         or None if the files are identical.
51     """
52
53     if os.path.isdir(file1) or os.path.isdir(file2):
54         return
55
56     html_output = ""
57
58     try:
59
60         with open(file1, 'r') as f1, open(file2, 'r') as f2:
61             # Read content of files line by line
62             data1 = f1.readlines()
63             data2 = f2.readlines()
64
65             # Use difflib to compare the lines
66             diff = difflib.unified_diff(data1, data2, fromfile=file1, tofile=file2)
67
68             if not diff:
69                 return None # Files are identical
70
71             # Generate basic HTML structure
72             html_output = """
73             <!DOCTYPE html>
74             <html>
75                 <head>
76                     <style>
77                         .added {{ background-color: lightgreen; }}
78                         .removed {{ background-color: lightcoral; }}
79                     </style>
80                 </head>
81                 <body>
82                     <h2>Differences between {} vs. {}</h2>
```

```

83     <pre>""".format(file1[len(workspace):], file2[len(workspace):])
84
85     # Add each line with highlighting
86     for line in diff:
87         if line.startswith("+"):
88             html_output += "<span class='added'>{}</span>\n".format(line.strip()
89                                         ())
90         elif line.startswith("-"):
91             html_output += "<span class='removed'>{}</span>\n".format(line.
92                                         strip())
93         else:
94             html_output += "{}\n".format(line.strip())
95
96     # Close HTML tags
97     html_output += "</pre></body></html>"
98
99
100    return html_output
101
102 def read_directory_contents(directory):
103     """
104     Reads the contents of a directory and its subdirectories recursively.
105
106     Args:
107         directory: Path to the directory to read.
108
109     Returns:
110         A list of full paths to all files within the directory and subdirectories.
111     """
112
113     all_files = []
114     for root, dirs, files in os.walk(directory):
115         for file in files:
116             # if file != '.git':
117             # Construct the full path to the file
118             full_path = os.path.join(root, file)
119             if not os.path.isdir(full_path):
120                 all_files.append(full_path)
121
122     return all_files
123
124 def switch_project_path(path, dest_proj):
125     parsed_path = path.split('/')
126     parsed_path[4] = dest_proj
127     return '/'.join(parsed_path)
128
129 def create_dir_for_path_to_file(path):
130     parsed_path = path.split('/')
131     dir_only_path = '/'.join(parsed_path[:-1])
132     try:
133         os.makedirs(dir_only_path) # Create directory and any missing parent
134                                         # directories
135     except FileExistsError:
136         pass # Ignore if the directory already exists
137
138 def remove_outputs(workspace, out):
139     try:
140         # Use shutil.rmtree with the ignore_errors argument

```

```

139     shutil.rmtree(workspace + out, ignore_errors=True)
140     print(f"Directory deleted successfully: {workspace+out}")
141 except OSError as e:
142     raise OSError(f"Error deleting directory: {workspace+out}. Reason: {e}")
143     from e
144
145 def compare_and_print(src, tgt, ref, workspace, out):
146     remove_outputs(workspace, out)
147     processed_files = []
148     for f in read_directory_contents(ref): # load list of customized aosp files
149         # create path/to/dir
150         if not os.path.isdir(f):
151             output_path = switch_project_path(f, 'diff/' + out)
152             create_dir_for_path_to_file(output_path)
153             diff = compare_files(switch_project_path(f, src.split('/')[4]),
154             switch_project_path(f, tgt.split('/')[4]), out)
155
156             with open(output_path+"_diff.html", "w") as f:
157                 if diff:
158                     if diff is not "":
159                         f.write(diff)
160                     else:
161                         f.write("""
162                             <!DOCTYPE html>
163                             <html>
164                             <head>
165                             <style>
166                             .added {{ background-color: lightgreen; }}
167                             .removed {{ background-color: lightcoral; }}
168                             </style>
169                             </head>
170                             <body>No Differences between {} vs. {}</h2>
171                             </body></html>""".format(switch_project_path(f, src.
172                                         split('/')[4]), switch_project_path(f, tgt.split(
173                                         '/')[4])))
174             processed_files.append(output_path)
175
176     with open(workspace + out + "/index.html", "w") as f:
177         f.write(html_prep(processed_files, src.split('/')[4] + " vs. " + tgt.split(
178                                         '/')[4] + " Differenced", workspace, out))
179
180 workspace = "/home/weiminn/Documents/diff/"
181
182 # they must follow aosp dir structure
183 aosp_de_dir = "/home/weiminn/Documents/customized_aosp" # used as reference on
184 # what to compare
185 aosp_13_dir = "/home/weiminn/Documents/aosp13"
186 aosp_14_dir = "/home/weiminn/Documents/aosp14"
187
188 # compare aosp13 vs aosp13 debloated
189 compare_and_print(aosp_13_dir, aosp_de_dir, aosp_de_dir, workspace, "13vs13debloat")
190
191 # compare aosp13 vs aosp14
192 compare_and_print(aosp_13_dir, aosp_14_dir, aosp_de_dir, workspace, "13vs14")

```

Understand Customized AOSP

```

art_quick_invoke_static_stub1
    art_quick_generic_jni_trampoline2
    artQuickGenericJniTrampoline3
    ART Execution4
    Forking process, binding application, launch activity5

```

5.2 March 14-30, 2024

5.2.1 Migrate Zicheng's code

Customize AOSP 14

file	Customized AOSP13	Changes in 14
java_vm_ext.cc	Modify FindSymbol.	No changes for FindSymbol.
art_method.h	Modify UpdateCounter declaration.	No changes for FindSymbol.
art_method-inl.h	Modify UpdateCounter definition (added unused variable declarations)	No changes for FindSymbol.
art_method.cc	Modify Invoke, and PrettifyMethod.	No changes for Invoke.
app_info.h	Add GetPackageName public declaration.	No related changes.
app_info.cc	Add GetPackageName definition.	No related changes.
runtime.h	Add NativeLibFunc struct, public declaration for a lot of functions and variables.	No related changes.

Table 5.2: Customizing AOSP 14

Scripts

Script to run the emulator and log console outputs:

¹<https://zhuanlan.zhihu.com/p/521498157>

²<https://juejin.cn/post/7249288285809524773>

³https://blog.csdn.net/sinat_38172893/article/details/74612596

⁴https://blog.csdn.net/sinat_38172893/article/details/72934122

⁵<https://medium.com/android-news/android-application-launch-explained-from-zygote-to-your-activity-1033a2a2a21>

```

1 emulator &
2 adb wait-for-device
3 A=$(adb shell getprop sys.boot_completed | tr -d '\r')
4
5 while [ "$A" != "1" ]; do
6     sleep 1
7     A=$(adb shell getprop sys.boot_completed | tr -d '\r')
8 done
9
10 adb root
11 echo "[WEIMINN] Enabling logging"
12 adb shell setprop debug.ld.all dlerror,dlopen
13 rm -f logcat.txt
14 echo "[WEIMINN] Logging with Logcat to logcat.txt"
15 adb logcat >> logcat.txt

```

Script to kill running emulator process:

```

1 adb devices | grep emulator | cut -f1 | while read line; do adb -s $line emu kill;
done

```

Script for finding a string:

```

1 cd /home/weiminn/Documents/aosp14/
2
3 clear
4
5 find \
6     ./frameworks/ ./art/ ./bionic/ \
7     \(-iname '*.java' -o -iname '*.cpp' -o -iname '*.cc' -o -iname '*.hpp' -o -
8     -iname '*.h' -o -iname '*.S' \) \
9     -a ! -iname '*test*' -a ! -iname '*out*' \
| xargs grep --color -sin 'invoke'

```

Comment to annotate added code:

```

1 /////////////////////////////////////////////////////////////////// DYNNDROID-START ///////////////////////////////////////////////////////////////////
2 /////////////////////////////////////////////////////////////////// DYNNDROID-END ///////////////////////////////////////////////////////////////////

```

Problems

Faced problem in runtime.h after copying in

```

1 void setLoadingSchema(bool loading, std::string s){
2     ALOGW("[Zicheng_Content_Provider] runtime.h::setLoadingSchema()::package:%s, %s"
3           , s.c_str(), loading ? "true" : "false");
4     loading_schema_ = loading;
5 }
6 void setLoadedSchema(bool loaded, std::string s){
7     ALOGW("[Zicheng_Content_Provider] runtime.h::setLoadedSchema()::package:%s", s.
8           c_str());
9     loaded_schema_ = loaded;
10 }
11 void setShouldReadSchema(bool should, std::string s){
12     ALOGW("[Zicheng_Content_Provider] runtime.h::setShouldReadSchema()::package:%s, %s"
13           , s.c_str(), should ? "true" : "false");

```

```
13     should_read_schema_ = should;
14 }
```

because there is no definition for ALOGW, so I put `#include "utils/Log.h"` at the top of `runtime.h`, and it worked!

Need to edit and compile `runtime.h` and `runtime.cc` after editing `art_method.cc`, as it calls `Runtime:::MYmatch_target_method`.

Changes to void `Instrumentation::InitializeMethodsCode` by both DynDroid and AOSP 14.

What does `.clang-format` mean?

5.3 March 31, 2024

5.3.1 Troubleshoot DynDroid on AOSP 14

Integrate Changes to Zygote code

Put customized code into remaining `dalvik_systemZygoteHooks.cc`, `odrefresh.cc`, and `odrefresh_main.cc`.

Test Run Minima in Emulator

Android is still crashing when using `adb install` command:

```
1 adb install success_apps/53_com.technoskill.cricketscore.apk
2 Performing Streamed Install
3 adb: failed to install success_apps/53_com.technoskill.cricketscore.apk: cmd:
      Failure calling service package: Broken pipe (32)
```

Test Package Manager in AOSP14 Clean

AOSP 14 Clean on P7Pro works for `adb install`.

Some have incompatible API versions:

```
1 adb install success_apps/68_im.dev.pregnancy.apk
2 Performing Streamed Install
3 adb: failed to install success_apps/68_im.dev.pregnancy.apk: Failure [
      INSTALL_FAILED_DEPRECATED_SDK_VERSION: App package must target at least SDK
      version 23, but found 16]
```

Preload Minima and Test Apps on Clean AOSP 14

Cannot build anymore:

```
1 internal error: bazel command failed: fork/exec ./build/bazel/bin/bazel: no such
  file or directory
```

Successfully preloaded MedicalDrugDictionary.apk, but the app crashes on load. Deleted out directory, and successfully reran the make command.

Weird problem where fastboot -w flashall does not remove the apps from previous version of OS. So, just do fastboot -w and it'll wipe properly.

Somehow preloading APKs causes them to crash at load:

```
1 03-31 15:04:36.501 4596 4596 W System.err: java.io.StreamCorruptedException:  
    invalid stream header: 4DF493E3  
2 03-31 15:04:36.501 4596 4596 W System.err: at java.io.ObjectInputStream.  
    readStreamHeader(ObjectInputStream.java:865)  
3 03-31 15:04:36.501 4596 4596 W System.err: at java.io.ObjectInputStream.<init>(  
    ObjectInputStream.java:356)  
4 03-31 15:04:36.501 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.a.b(Unknown Source:10)  
5 03-31 15:04:36.501 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.a.c(Unknown Source:2)  
6 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.a.b(Unknown Source:4)  
7 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.a.<init>(Unknown Source:12)  
8 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.a.a(Unknown Source:6)  
9 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.  
    dictionary.DrugsListActivity.onCreate(Unknown Source:21)  
10 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Activity.  
    performCreate(Activity.java:8621)  
11 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Activity.  
    performCreate(Activity.java:8599)  
12 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Instrumentation.  
    callActivityOnCreate(Instrumentation.java:1456)  
13 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.  
    performLaunchActivity(ActivityThread.java:3804)  
14 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.  
    handleLaunchActivity(ActivityThread.java:3963)  
15 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.  
    LaunchActivityItem.execute(LaunchActivityItem.java:103)  
16 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.  
    TransactionExecutor.executeCallbacks(TransactionExecutor.java:139)  
17 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.  
    TransactionExecutor.execute(TransactionExecutor.java:96)  
18 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread$H.  
    handleMessage(ActivityThread.java:2468)  
19 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Handler.  
    dispatchMessage(Handler.java:106)  
20 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Looper.loopOnce(Looper.  
    java:205)  
21 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Looper.loop(Looper.  
    java:294)  
22 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.main(  
    ActivityThread.java:8248)  
23 03-31 15:04:36.502 4596 4596 W System.err: at java.lang.reflect.Method.invoke(  
    Native Method)  
24 03-31 15:04:36.502 4596 4596 W System.err: at com.android.internal.os.  
    RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:552)  
25 03-31 15:04:36.502 4596 4596 W System.err: at com.android.internal.os.  
    ZygoteInit.main(ZygoteInit.java:971)  
26 03-31 15:04:36.502 4596 4596 W System.err: java.io.StreamCorruptedException:
```

```
        invalid stream header: 35BBE049
27 03-31 15:04:36.502 4596 4596 W System.err: at java.io.ObjectInputStream.
     readStreamHeader(ObjectInputStream.java:865)
28 03-31 15:04:36.502 4596 4596 W System.err: at java.io.ObjectInputStream.<init>(
     ObjectInputStream.java:356)
29 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.a.b(Unknown Source:10)
30 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.a.c(Unknown Source:10)
31 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.a.b(Unknown Source:4)
32 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.a.<init>(Unknown Source:12)
33 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.a.a(Unknown Source:6)
34 03-31 15:04:36.502 4596 4596 W System.err: at com.atomic.apps.medical.drug.
     dictionary.DrugsListActivity.onCreate(Unknown Source:21)
35 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Activity.
     performCreate(Activity.java:8621)
36 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Activity.
     performCreate(Activity.java:8599)
37 03-31 15:04:36.502 4596 4596 W System.err: at android.app.Instrumentation.
     callActivityOnCreate(Instrumentation.java:1456)
38 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.
     performLaunchActivity(ActivityThread.java:3804)
39 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.
     handleLaunchActivity(ActivityThread.java:3963)
40 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.
     LaunchActivityItem.execute(LaunchActivityItem.java:103)
41 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.
     TransactionExecutor.executeCallbacks(TransactionExecutor.java:139)
42 03-31 15:04:36.502 4596 4596 W System.err: at android.app.servertransaction.
     TransactionExecutor.execute(TransactionExecutor.java:96)
43 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread$H.
     handleMessage(ActivityThread.java:2468)
44 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Handler.
     dispatchMessage(Handler.java:106)
45 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Looper.loopOnce(Looper
     .java:205)
46 03-31 15:04:36.502 4596 4596 W System.err: at android.os.Looper.loop(Looper.
     java:294)
47 03-31 15:04:36.502 4596 4596 W System.err: at android.app.ActivityThread.main(
     ActivityThread.java:8248)
48 03-31 15:04:36.502 4596 4596 W System.err: at java.lang.reflect.Method.invoke(
     Native Method)
49 03-31 15:04:36.502 4596 4596 W System.err: at com.android.internal.os.
     RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:552)
50 03-31 15:04:36.502 4596 4596 W System.err: at com.android.internal.os.
     ZygoteInit.main(ZygoteInit.java:971)
51 03-31 15:04:36.502 4596 4596 D com.atomic.apps.medical.drug.dictionary: Using B
52 03-31 15:04:36.502 4596 4596 D AndroidRuntime: Shutting down VM
53 03-31 15:04:36.502 4596 4596 E AndroidRuntime: FATAL EXCEPTION: main
54 03-31 15:04:36.502 4596 4596 E AndroidRuntime: Process: com.atomic.apps.medical.
     drug.dictionary, PID: 4596
55 03-31 15:04:36.502 4596 4596 E AndroidRuntime: java.lang.RuntimeException:
     Unable to start activity ComponentInfo{com.atomic.apps.medical.drug.dictionary
     /com.atomic.apps.medical.drug.dictionary.DrugsListActivity}: java.lang.
     NullPointerException: Attempt to invoke virtual method 'byte[] java.lang.
     String.getBytes()' on a null object reference
```

```

56 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.ActivityThread.
      performLaunchActivity(ActivityThread.java:3822)
57 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.ActivityThread.
      handleLaunchActivity(ActivityThread.java:3963)
58 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.servertransaction
      .LaunchActivityItem.execute(LaunchActivityItem.java:103)
59 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.servertransaction
      .TransactionExecutor.executeCallbacks(TransactionExecutor.java:139)
60 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.servertransaction
      .TransactionExecutor.execute(TransactionExecutor.java:96)
61 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.ActivityThread$H.
      handleMessage(ActivityThread.java:2468)
62 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.os.Handler.
      dispatchMessage(Handler.java:106)
63 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.os.Looper.loopOnce(
      Looper.java:205)
64 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.os.Looper.loop(Looper
      .java:294)
65 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.ActivityThread.
      main(ActivityThread.java:8248)
66 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at java.lang.reflect.Method.
      invoke(Native Method)
67 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.android.internal.os.
      RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:552)
68 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.android.internal.os.
      ZygoteInit.main(ZygoteInit.java:971)
69 03-31 15:04:36.502 4596 4596 E AndroidRuntime: Caused by: java.lang.
      NullPointerException: Attempt to invoke virtual method 'byte[] java.lang.
      String.getBytes()' on a null object reference
70 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.atomic.apps.medical.drug.
      dictionary.a.a(Unknown Source:0)
71 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.atomic.apps.medical.drug.
      dictionary.a.b(Unknown Source:8)
72 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.atomic.apps.medical.drug.
      dictionary.a.<init>(Unknown Source:12)
73 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.atomic.apps.medical.drug.
      dictionary.a.a(Unknown Source:6)
74 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at com.atomic.apps.medical.drug.
      dictionary.DrugsListActivity.onCreate(Unknown Source:21)
75 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.Activity.
      performCreate(Activity.java:8621)
76 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.Activity.
      performCreate(Activity.java:8599)
77 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.Instrumentation.
      callActivityOnCreate(Instrumentation.java:1456)
78 03-31 15:04:36.502 4596 4596 E AndroidRuntime: at android.app.ActivityThread.
      performLaunchActivity(ActivityThread.java:3804)
79 03-31 15:04:36.502 4596 4596 E AndroidRuntime: ... 12 more
80 03-31 15:04:36.504 1496 2267 W ActivityTaskManager: Force finishing activity
      com.atomic.apps.medical.drug.dictionary/.DrugsListActivity

```

Replace Framework and Run

PackageManager.java, DefaultPermissionGrantPolicy.java, PermissionManagerServiceImpl.java, ParsingPackageUtils.java has only trivial logging code.

PermissionManagerService.java has some changes to requested permissions in

onPackageInstalled. PackageParser.java has changes to parseBaseApkCommon.

Same problem with emulator version:

```

1 adb install success_apps/60_com.atomic.apps.medical.drug.dictionary.apk
2 Performing Streamed Install
3 adb: failed to install success_apps/60_com.atomic.apps.medical.drug.dictionary.apk
   : cmd: Failure calling service package: Broken pipe (32)

```

Logcat output:

```

1 03-31 15:47:11.367 6659 6709 E AndroidRuntime: *** FATAL EXCEPTION IN SYSTEM
   PROCESS: PackageManager
2 03-31 15:47:11.367 6659 6709 E AndroidRuntime: java.lang.
   UnsupportedOperationException
3 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at java.util.
   Collections$UnmodifiableCollection.add(Collections.java:1114)
4 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   permission.PermissionManagerService$PermissionManagerServiceInternalImpl.
   onPackageInstalled(PermissionManagerService.java:706)
5 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallPackageHelper.updateSettingsInternal(InstallPackageHelper.java:2370)
6 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallPackageHelper.updateSettingsLI(InstallPackageHelper.java:2165)
7 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallPackageHelper.commitPackagesLocked(InstallPackageHelper.java:2134)
8 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallPackageHelper.installPackagesLI(InstallPackageHelper.java:1031)
9 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallPackageHelper.installPackagesTraced(InstallPackageHelper.java:915)
10 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallingSession.processApkInstallRequests(InstallingSession.java:540)
11 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallingSession.processInstallRequests(InstallingSession.java:529)
12 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallingSession.lambda$processPendingInstall$0(InstallingSession.java:288)
13 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallingSession.$r8$lambda$tqRjKCgCJYNNnnY7Qw5M5BHLup8(InstallingSession.
   java:0)
14 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.pm.
   InstallingSession$$ExternalSyntheticLambda1.run(R8$$SyntheticClass:0)
15 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at android.os.Handler.
   handleCallback(Handler.java:958)
16 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at android.os.Handler.
   dispatchMessage(Handler.java:99)
17 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at android.os.Looper.loopOnce(
   Looper.java:205)
18 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at android.os.Looper.loop(Looper
   .java:294)
19 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at android.os.HandlerThread.run(
   HandlerThread.java:67)
20 03-31 15:47:11.367 6659 6709 E AndroidRuntime: at com.android.server.
   ServiceThread.run(ServiceThread.java:46)

```

Problematic code:

```

1 String name = "edu.smu.minimaconfig.MyContentProvider.READ";
2 int index_1 = pkg.getRequestedPermissions().indexOf(name);
3 if (index_1 == -1) {
4     pkg.getRequestedPermissions().add(name.intern());

```

```
5     Slog.w("zicheng_framework", "[Zicheng_frameworks]onPackageInstalled() add
6         permission:"
7     + name + " in package: " + pkg.getPackageName());
8 } else {
9     Slog.w("zicheng_framework", "[Zicheng_frameworks]onPackageInstalled() Ignoring
10    duplicate uses-permissions/uses-permissions-sdk-m: "
11    + name + " in package: " + pkg.getPackageName());
12 }
13 if(pkg.getRequestedPermissions().size()>0)
14     for(int kk=0;kk<pkg.getRequestedPermissions().size();kk++)
15         Slog.w("zicheng_framework", "[Zicheng_frameworks]PermissionManagerService .
16             java onPackageInstalled()"+
17             " in package: " + pkg.getPackageName() + "permission num: "+kk
18             +": "+pkg.getRequestedPermissions().get(kk));
```

Removed this line of code from emulator project also. The app can be successfully adb installed but the app crashes after loading splash screen for a long time.

Chapter 6

April

6.1 April 01, 2024

6.1.1 Complete Lifecycle Hook Execution Flow

Android Lifecycle Hooks

onCreate, onStart, onPause, onStop, onResume, onDestroy¹

onCreate Hook Ran by ActivityThread

Look for `ActivityThread.java` which has `performLaunchActivity` method which calls `onCreate`²

It is called by `ClientTransactionHandler`'s `handleLaunchActivity` which is called by³

What's the difference between `BIND_APPLICATION` and `attachApplication`?

`BIND_APPLICATION` is the message handled by `ActivityThread` which goes on to call `attachApplication`.

What's the difference between Launch Activity vs Start Activity?

`BIND_APPLICATION`⁴ message is sent to the system server's looper after new process is spawned, to bind the application to the process.

¹<https://developer.android.com/guide/components/activities/activity-lifecycle>

²<https://www.jianshu.com/p/c8e09bc142fa>

³<https://blog.csdn.net/shulianghan/article/details/120263706>

⁴<https://medium.com/android-news/android-application-launch-explained-from-zygote-to-your-activity-oncreate-8a8ff>

LAUNCH_ACTIVITY is another message after binding the application to launch the main activity of the application ⁵⁶⁷⁸⁹¹⁰

How does Zygote call ActivityThread's main?

ZygoteInit's main calls ActivityThread's constructor which prepares the main looper where all app events are executed¹¹.

runp.sh for Building, Flashing and Logging Pipeline

```

1  adb reboot bootloader
2
3  echo "[WEIMINN] Wiping data"
4  fastboot erase userdata
5
6  echo "[WEIMINN] Building AOSP"
7  cd /media/weimin/Downloads/aosp14new
8  source build/envsetup.sh
9  lunch aosp_cheetah-userdebug
10 make
11
12 echo "[WEIMINN] Flashing ROM"
13 flash_start='date +%s'
14 fastboot -w flashall
15 flash_end='date +%s'
16 echo Flash time was `expr $flash_end - $flash_start` seconds.
17
18 boot_start='date +%s'
19 adb wait-for-device
20
21 echo "[WEIMINN] Waiting for boot completion"
22 A=$(adb shell getprop sys.boot_completed | tr -d '\r')
23 while [ "$A" != "1" ]; do
24 echo "[WEIMINN] Waiting for boot completion"
25 sleep .5
26 A=$(adb shell getprop sys.boot_completed | tr -d '\r')
27 done
28 boot_end='date +%s'
29 echo Boot time was `expr $boot_end - $boot_start` seconds.
30
31 adb root
32
33 echo "[WEIMINN] Enabling logging"
34 adb shell setprop debug.ld.all dlerror,dlopen
35

```

⁵<https://blog.csdn.net/Tenderness4/article/details/83353636>

⁶<https://juejin.cn/post/7005062882966634526>

⁷https://www.zhihu.com/tardis/zm/art/468648584?source_id=1003

⁸<https://sleticalboy.github.io/android/2019/04/13/app%E5%90%AF%E5%8A%A8%E6%B5%81%E7%A8%8B%E5%88%86%E6%90-02/>

⁹<https://sleticalboy.github.io/assets/android/android-app-start.svg>

¹⁰<https://dev.to/pyricau/android-vitals-how-adb-measures-app-startup-5n7>

¹¹<https://stackoverflow.com/questions/46300145/what-the-matter-between-zygote-and-ui-thread-of-an>

```

36 rm -f logcat.txt
37 echo "[WEIMINN] Logging with Logcat to logcat.txt"
38 adb logcat >> logcat.txt

```

6.2 April 02-07, 2024

6.2.1 Complete ART Execution Flow

How is Java Code loaded into Android Runtime?

Deep Dive into ART¹², and Method Execution Flow¹³, LinkCode¹⁴¹⁵, Loading OAT files¹⁶, Executing Methods¹⁷, Virtual Machine¹⁸, Loading Classes and Methods¹⁹

Loading OAT Files

An Application has only one `classes.dex` file which is compiled into an OAT file via LLVM compiler and stored in the `oatexec` section of the OAT file. It is also possible to load DEX file during run time which are also translated into OAT.

Loading OAT Methods

`FindClass` in `jni_internal.cc` accesses singleton instance of the Runtime, singleton instance of `ClassLinker`, and `ClassLoader` instance. And the pointer to the `ClassLoader` instance is passed into the `FindClass` function of `class_linker.cc`.

6.3 April 08, 2024

6.3.1 Meeting with iSprint

Minutes

Attendees: Prof. Shar, Albert, Vikas, Phong Phan, Yan Naing, Wei Minn

1. Albert's agenda: 1) Publication of research on this discovered malware, and 2) Discuss plans for the next iteration of Shieldcon this year.

¹²<https://www.youtube.com/watch?v=mFq0vNvUgj8>

¹³https://huanle19891345.github.io/en/android/art/jni/java_jni%E6%96%B9%E6%B3%95%E8%B0%83%E7%94%A8%E5%8E%9F%E7%90%86/

¹⁴<https://cloud.tencent.com/developer/article/2343052>

¹⁵<https://blog.csdn.net/luoshengyang/article/details/39533503>

¹⁶<https://blog.csdn.net/luoshengyang/article/details/39307813>

¹⁷<https://blog.csdn.net/Luoshengyang/article/details/40289405>

¹⁸<https://blog.csdn.net/luoshengyang/article/details/8914953>

¹⁹<https://blog.csdn.net/luoshengyang/article/details/39533503>

2. Albert would like to continue holding Shieldcon with SMU; he is eager to find a better venue than the previous time; the exact timing of the conference depends on the finding of the malware; he plans to leverage publicity of the conference.

3. Albert mentioned the preliminary survey conducted on the banks regarding measures for the vulnerabilities in their apps. Many banks did not respond to this malware attack.

4. Phong presents FjordPhantom, virtualization-based malware. He demonstrated a container can run on any phone and requires no root no root required. It utilizes virtualization to run a container that wraps a malware that hooks the target app that looks identical to bank app

5. This type of malware is very flexible and module; can easily repackage any android app.

6. Albert Chings said they discovered another similar malware a few weeks ago; but still conducting experiments to confirm.

7. This malware relies on sideloading of APK into your Android phone.

8. Phong's Demo of FjordPhantom on emulator:

a. Install demo bank APK via sideloading

b. Install screenreader app via sideloading

c. Demo bank app warns of enabled accessibility, and do not allow you to proceed

d. Inject container inside the app using a Python script and re-sideload the now tampered OCBC app

e. With modified OCBC app, even with screenreader and accessibility turned on, there is no warning message, and the user can proceed.

9. The container also contains Hooking framework which plays essential role in this attack.

10. Albert asserted that more than 80% of the bank apps are susceptible to this attack

11. Albert blames this vulnerability on poorly written apps.

12. Albert calls for more research and warn the public about this malware.

13. Prof. Shar aims for ASE conference which will be around July.

14. Vikas asked about the company that first detected this malware at the end of 23, and inquired about the differences between their release and i-Sprint's plans; Albert confirmed that they are partners with the company that discovered it.

15: Albert said he is planning an extension to last year's paper, and asked Prof. Shar of any better way to proceed.

16: Prof. Shar plans on focusing on this particular attack, rather than as a continuation of last year's work.

17: Albert suggested two separate papers: 1) this attack, and 2) progress of previous work one year later.

18: Prof. Shar insists to focus on attack for the time being.

19: Albert shares about the concerns that CSA and other agencies across the region

have about this attack; he also talked about the potential for more publicity for i-Sprint and SMU regarding the release of this malware.

20: Prof. Shar asked Albert on how much information can be disclose in the paper with regards to the banks, and the details of the attacks?

21: Albert replies that we cannot divulge the affected banks' names, and instead can follow the press releases of i-Sprint. He added that the banks have already been warned.

22: Prof. Shar assigns Wei Minn to take the lead for this paper.

23. After brief discussion about the Shieldcon date, we agreed to meet up again in person for more details like last time.

24: Prof. Shar asked for 1 or 2 weeks to get back to i-Sprint.

25: Albert, Phong and Vikas leaves Meeting

26: Prof. Shar tasked Wei Minn with setting the skeleton for the paper (follow Vikas' example from last time) and get David involved only toward the end.

6.4 April 09-15, 2024

6.4.1 Recompile AOSP 13

Why is it not logging to Logcat?

Log is not showing, even if `LOG_NDEBUG` is set to 0 in `AndroidRuntime.cpp`.

Okay, now it works and I didn't edit anything further.

6.4.2 Which components are involved in ART?

Finishing up the sequence flow.

6.4.3 Components of Debloater Flow in AOSP 13

ART Procedures in Debloating Flow

File	Procedure
<code>libartbase</code>	
<code>os_linux.h</code>	Add declaration <code>CreateMINIMAFfile</code>
<code>os_linux.h</code>	Add declaration <code>CreateDirectories2</code>
<code>os_linux.cc</code>	Add definition <code>CreateMINIMAFfile</code>
<code>os_linux.cc</code>	Add definition <code>CreateDirectories2</code>
<code>libdexfile</code>	
<code>art_dex_file_loader.cc</code>	Modify definition <code>ArtDexFileLoader::Open</code>

File	Procedure
<code>dex_file_loader.cc</code>	Modify definition <code>DexFileLoader::OpenCommon</code>
<code>libnativeloader</code>	
<code>native_loader_namespace.cpp</code>	Modify definition <code>NativeLoaderNamespace::Load</code>
<code>native_loader.cpp</code>	Modify definition <code>OpenNativeLibrary</code>
<code>native_loader.cpp</code>	Modify definition <code>OpenNativeLibraryInNamespace</code>
<code>oderefresh</code>	
<code>oderefresh.cc</code>	Modify definition <code>AddDex2OatProfileAndCompilerFilter</code>
<code>oderefresh_main.cc</code>	Modify definition <code>InitializeConfig</code>
<code>runtime</code>	
<code>app_info.h</code>	Add declaration <code>GetPackageName</code>
<code>app_info.cc</code>	Modify definition <code>GetPackageName</code>
<code>art_method-inl.h</code>	Modify declaration
	<code>ArtMethod::CheckIncompatibleClassChange</code>
<code>art_method-inl.h</code>	Modify declaration
	<code>ArtMethod::UpdateCounter</code>
<code>art_method.h</code>	Modify declaration <code>ArtMethod::UpdateCounter</code>
<code>art_method.cc</code>	Modify declaration <code>ArtMethod::Invoke</code>
<code>art_method.cc</code>	Modify declaration <code>ArtMethod::PrettyMethod</code>
<code>class_linker.cc</code>	Modify declaration <code>ClassLinker::FindClass</code>
<code>class_linker.cc</code>	Modify declaration <code>LinkCode</code>
<code>class_linker.cc</code>	Modify declaration <code>ClassLinker::LoadClass</code>
<code>instrumentation.cc</code>	Modify declaration <code>CanUseNterp</code>
<code>instrumentation.cc</code>	Modify declaration <code>Instrumentation::InitializeMethodsCode</code>
<code>oat_file_manager.cc</code>	Modify declaration <code>OatFileManager::OpenDexFilesFromOat_Impl</code>
<code>runtime.h</code>	Add declaration <code>struct NativeLibFunc;</code>
<code>runtime.h</code>	Add declaration <code>MYmatch_hook_method</code>
<code>runtime.h</code>	Add declaration <code>MYmatch_target_method</code>
<code>runtime.h</code>	Add declaration <code>MYreadFile</code>
<code>runtime.h</code>	Add declaration <code>MYmake_file</code>
<code>runtime.h</code>	Add declaration <code>read_content</code>
<code>runtime.h</code>	Add definition <code>getLoadingSchema</code>
<code>runtime.h</code>	Add definition <code>getLoadedSchema</code>
<code>runtime.h</code>	Add definition <code>getShouldReadSchema</code>
<code>runtime.h</code>	Add definition <code>setLoadingSchema</code>
<code>runtime.h</code>	Add definition <code>setLoadedSchema</code>

File	Procedure
runtime.h	Add definition setShouldReadSchema
runtime.h	Add definition getDebugging
runtime.h	Add definition getTarget_method_string_vector
runtime.h	Add definition setTarget_method_string_vector
runtime.h	Add definition getMINIMA_file_path
runtime.h	Add definition getMINIMA_folder_path
runtime.h	Add definition getHookMethodName
runtime.h	Add declaration MINIMA_file_path
runtime.h	Add declaration hook_method_name
runtime.h	Add declaration target_method_string_vector
runtime.h	Add declaration target_native_func_string_vector
runtime.h	Add declaration debugging
runtime.h	Add declaration api_prefix_List
runtime.h	Add declaration loaded_schema_
runtime.h	Add declaration loading_schema_
runtime.h	Add declaration should_read_schema_
runtime.cc	Add definition StringAppendV
runtime.cc	Add definition StringPrintf
runtime.cc	Add definition StringAppendF
runtime.cc	Add definition Runtime::MINIMA_file_path
runtime.cc	Add definition Runtime::hook_method_name
runtime.cc	Add definition Runtime::debugging
runtime.cc	Add definition
runtime.cc	Runtime::target_method_string_vector
runtime.cc	Add definition
runtime.cc	Runtime::target_native_func_string_vector
runtime.cc	Add definition api_prefix_List
runtime.cc	Add definition NativeLibFunc
runtime.cc	Modify definition Runtime
runtime.cc	Modify definition Create
runtime.cc	Modify definition Start
runtime.cc	Modify definition Init
runtime.cc	Add definition MYmatch_hook_method
runtime.cc	Add definition java2cpp
runtime.cc	Add definition is_within_JNImethod_list
runtime.cc	Add definition Runtime::read_content
runtime.cc	Add definition is_packageMethod
runtime.cc	Add definition Runtime::MYmatch_target_method
runtime.cc	Add definition Runtime::MYmake_file
runtime.cc	Add definition Runtime::MYreadFile

File	Procedure
trace.cc	Add definition modifyFilePermissions
trace.cc	Modify definition Trace::Start
trace.cc	Modify definition Trace::Trace
trace.cc	Modify definition Trace::MethodEntered
trace.cc	Modify definition Trace::MethodExited
runtime/entrypoints	
quick_trampoline_entry_points.cc	Modify definition artQuickToInterpreterBridge
runtime/interpreter	
interpreter_common.cc	Modify definition DoCallCommon
interpreter_switch_impl-inl.h	Modify definition ExecuteSwitchImplCpp
interpreter_switch_impl.h	Modify definition ExecuteSwitchImpl
interpreter.cc	Modify definition Execute
interpreter.cc	Modify definition EnterInterpreterFromEntryPoint
runtime/jit	
jit.cc	Modify definition MethodEntered
runtime/jni	
java_vm_ext.cc	Modify definition FindSymbol
java_vm_ext.cc	Modify definition JavaVMExt::LoadNativeLibrary
jni_internal.cc	Modify definition FindClass
runtime/mirror	
class.cc	Modify definition FindClassMethodWithSignature
runtime/native	
dalvik_system_DexFile.cc	Modify definition DexFile_openDexFileNative
dalvik_system_ZygoteHooks.cc	Add definition modifyFilePermissions1
dalvik_system_ZygoteHooks.cc	Modify definition ZygoteHooks_nativePostForkChild

Table 6.1: ART Procedures involved in Debloating

Bionic Procedures in Debloating Flow

File	Procedure
libc	
execinfo/include/bionic/execinfo.h	Delete everything and add ../../../../include/execinfo.h
fts/include/bionic/fts.h	Delete everything and add ../../../../include/fts.h
libdl	
libdl.cpp	Modify definition android_dlopen_ext
linker	
delfcn.cpp	Modify definition dlopen_ext
delfcn.cpp	Modify definition __loader_android_dlopen_ext
linker_phdr.h	Add declaration add_de_lib_name_vec
linker_phdr.h	Add declaration add_de_func_name_vec
linker_phdr.h	Add declaration match_target_so_name
linker_phdr.h	Add declaration match_target_func_name
linker_phdr.h	Add declaration dynsym_fragments
linker_phdr.h	Add declaration dynsym_table
linker_phdr.h	Add declaration de_dynsym_vec
linker_phdr.h	Add declaration de_func_name_vec
linker_phdr.h	Add declaration de_natlib_name_vec
linker_phdr.h	Add declaration handled_array
linker_phdr.cpp	Add definition debug_zicheng_native
linker_phdr.cpp	Modify definition ElfReader
linker_phdr.cpp	Modify definition ElfReader::Load
linker_phdr.cpp	Add definition compareDynsymPointers
linker_phdr.cpp	Add definition compareDynsyms
linker_phdr.cpp	Modify definition ElfReader::ReadProgramHeaders
linker_phdr.cpp	Modify definition ElfReader::ReadSectionHeaders
linker_phdr.cpp	Modify definition ElfReader::ReadDynamicSection
linker_phdr.cpp	Modify definition phdr_table_get_load_size
linker_phdr.cpp	Modify definition ElfReader::ReserveAddressSpace
linker_phdr.cpp	Modify definition ElfReader::LoadSegments
linker_phdr.cpp	Add definition

File	Procedure
linker_phdr.cpp	ElfReader::add_de_func_name_vec Add definition
linker_phdr.cpp	ElfReader::add_de_lib_name_vec Add definition
linker_phdr.cpp	ElfReader::match_target_so_name Add definition
linker.cpp	ElfReader::match_target_func_name Modify definition
linker.cpp	open_library_in_zipfile
linker.cpp	Modify definition find_libraries
linker.cpp	Modify definition find_library
linker.cpp	Modify definition do_dlopen

Table 6.2: Bionic Procedures involved in Debloating**Framework Procedures in Debloating Flow**

File	Procedure
base/core/java/android/content/pm PackageManager.java	Modify method getPackageArchiveInfo
PackageManager.java	Modify method generatePackageInfo
PackageParser.java	Modify method generatePackageInfo (overloaded)
PackageParser.java	Modify method parseBaseApkCommon
PackageParser.java	Modify method parseUsesPermission
services/core/java/com/android/ server/pm/permission DefaultPermissionGrantPolicy.java	Modify method grantRuntimePermissions
PermissionManagerService.java	Modify method onPackageInstalled
PermissionManagerServiceImpl.java	Add lot of trivial logging code
services/core/java/com/android/ server/pm/pkg/parsing ParsingPackageUtils.java	Modify method parseSplitBaseAppChildTags

Table 6.3: Framework Procedures involved in Debloating

6.4.4 Test Customized Code without the External Headers

There seems to be a group of files where the contents are emptied out and references another file instead: `bionic/libc/execinfo/include/bionic/execinfo.h`, `bionic/libc/fts/include/bionic/fts.h`, `bionic/tools/versioner/platforms/libc.map.txt`, `bionic/tools/versioner/current`, `bionic/tools/versioner/dependencies`, `bionic/libfdtrack/.clang-format`, and `bionic/.clang-format`. `boot-image-profile.txt` also has a lot of changes.

Now having problem: build system doesn't detect any changes after updating code. Deleted the out and recompile again. Now it works.

6.4.5 ART Loading Process of OAT Files

The process of ART start up is already been explored²⁰²¹. New thing contributed is the Application installation process by Package Manager Service.

Optimizing DEX into OAT files

Procedure	Description
<code>frameworks/base/services/core/java/com/android/server/pm/Installer.java</code>	
<code>dexopt</code>	Optimize DEX bytecode in the APK
<code>frameworks/native/cmds/installld/dexopt.cpp</code>	
<code>dexopt</code>	
<code>frameworks/native/cmds/installld/run_dex2oat.cpp</code>	
<code>RunDex2oat::RunDex2oat</code>	
<code>RunDex2oat::Initialize</code>	
<code>RunDex2oat::Exec</code>	

Table 6.4: APK Installation Process

Loading OAT

Procedure	Description
<code>art/runtime/runtime.cc</code>	

²⁰<https://blog.csdn.net/luoshengyang/article/details/18006645>

²¹<https://blog.csdn.net/luoshengyang/article/details/39307813>

Procedure	Description
Runtime::Create	Create ART virtual machine, which will be used to start Zygote Process which runs System Server and Android Apps.
Runtime::Init	Parses and parameters for the startup of ART (if image file (an OAT file containing multiple DEX files) is not specified, /system/framework/boot.art will be prepared in advance in the system partition to be used as ART virtual machine); create an ART Heap based on the parameters
art/runtime/gc/heap.cc Heap::Heap	
art/runtime/gc/space/image_space.cc ImageSpace:: LoadBootImage ImageSpace:: BootImageLoader:: LoadFromSystem ImageSpace:: BootImageLoader:: LoadImage ImageSpace:: BootImageLoader:: ReserveBootImageMemory	
art/libartbase/base/mem_map.cc MemMap::MapAnonymous	
art/runtime/gc/space/image_space.cc ImageSpace:: BootImageLoader:: LoadComponents ImageSpace:: Calls OatFile::Open BootImageLoader:: OpenOatFile	
art/runtime/oat_file.cc OatFile::Open	Called by ImageSpace::BootImageLoader:: OpenOatFile

Procedure	Description
OatFile:::OpenOatFile	Calls OatFileBase:::OpenOatFile template method based on DlOpenOatFile or ElfOatFile, both of which extends OatFileBase.
art/runtime/oat_file.cc	
OatFileBase:::Setup	OatFileBase extends OatFile. Called as template instance by OatFile:::OpenOatFile.
OatDexFile:::OatDexFile	New instance of OatDexFile:::OatDexFile created by OatFileBase:::Setup
GetOatHeader	
art/runtime/elf_file.cc	
ElfOatFile:::Load	Called as template instance by OatFile:::OpenOatFile if it implements ElfOatFile.
ElfOatFile::: ElfFileOpen ElfFile:::Open	
art/runtime/gc/space/image_space.cc	
ImageSpace::: BootImageLayout::: LoadFromSystem ImageSpace::: BootImageLayout::: LoadOrValidateFromSystem ImageSpace::: BootImageLayout::: LoadOrValidate	
art/runtime/runtime.cc	
JavaVMExt:::Create	Java VM instance for the caller to interact with the ART.
ClassLinker::: CreateFromCompiler	Called if the first continuous space in this list is not an Image space (current process is not a Zygote process, but a dex2oat process started when application is installed). To be used by ART for loading Java classes.
ClassLinker::: CreateFromImage	Called if the first continuous space in this list is an Image space. To be used by ART for loading Java classes.

Table 6.5: OAT Loading Process

6.5 April 16-29, 2024

6.5.1 ART Loading Process of Classes and Methods

Need to get the corresponding OAT class in the OAT file using the number of target DEX class.

Procedure	Description
frameworks/base/core/jni/AndroidRuntime.cpp	
AndroidRuntime::start	
AndroidRuntime::	
startVm	
art/runtime/jni/java_vm_ext.cc	
JNI_CreateJavaVM	
art/runtime/runtime.cc	
Runtime::Create	Before starting ART, you have to first create ART virtual machine, which will be used to start Zygote Process which runs System Server and Android Apps.
Runtime::Init	Parses and parameters for the startup of ART (if image file (an OAT file containing multiple DEX files) is not specified, /system/framework/boot.art will be prepared in advance in the system partition to be used as ART virtual machine); create an ART Heap based on the parameters
art/runtime/thread.h	
GetJniEnv	Get APIs to ART Java VM of current running thread. The APIs are wrapped in struct JNIEnvExt which is an extension of JNIEnv, and stores JNIInterface.
libnativehelper/include_jni/jni.h	
JNIEnv	Wraps functions to call ART Java VM to execute Java classes and methods.
JNIInterface	Wraps functions to call ART Java VM to execute Java classes and methods. Among them is JNI::FindClass which is defined in art/runtime/jni/jni_internal.cc.
art/runtime/jni/jni_internal.cc	

Procedure	Description
FindClass	Check if runtime instance has started, and then calls ClassLinker's FindClass. If not started, calls ClassLinker's FindSystemClass (which also calls ClassLinker) instead.
art/runtime/class_linker.cc	
ClassLinker::FindClass	
ClassLinker::	Check whether the class has already been loaded. If so, just return the corresponding class object.
LookupClass	
ClassLinker::	Find the class in DexFile, if class_loader is null.
FindInClassPath	Then call ClassLinker::DefineClass to load the class from the loaded DEX.
ClassLinker::	Use the class_loader object (of java.lang.ClassLoader type) to load the specified class.
DefineClass	
ClassLinker::LoadClass	Load the specified class from specified DEX file, and add it to the loaded class list of ClassLinker using InsertClass. Get the class index number of the loading class in the DEX file using GetIndexForClassDef to obtain OatClass structure in the corresponding OAT file through the class index number. Each member functions of the class being loaded will be instantiated as ArtMethod objects.
LinkCode	Determine if the loaded classes and methods are executed though the interpreter, or with local machine instructions.
art/runtime/instrumentation.cc	
Instrumentation::	
InitializeMethodsCode	
Instrumentation::	
UpdateMethodsCode	
Instrumentation::	
UpdateMethodsCodeImpl	
UpdateEntryPoints	
art/runtime/art_method.h	
SetEntryPointFromQuickCompiledCode	
SetEntryPointFromQuickCompiledCodePtrSize	
SetNativePointer	
art/runtime/jit/jit.cc	

Procedure	Description
EntryPointFromQuickCompiledCodeOffset	
art/runtime/jni/jni_env_ext.cc GetJniEnv	Get APIs to ART Java VM of current running thread. The APIs are wrapped in struct JNIEnv.
art/runtime/thread.h GetJniEnv	Called by JNI_CreateJavaVM

Table 6.6: Class Loading Process

Imagespace²²²³, and Heap²⁴

6.5.2 ART Execution Process of Classes and Methods

Procedure	Description
art/runtime/runtime.cc Runtime::Init	
art/runtime/thread.cc Thread::Attach Thread::Init Thread:: InitTlsEntryPoints	
art/runtime/arch/arm64/entrypoints_init_arm64.cc InitEntryPoints	
art/runtime/entrypoints/quick/quick_default_init_entrypoints.h DefaultInitEntryPoints	
art/runtime/class_linker.cc LinkCode	In AOSP 13 it doesn't call artInterpreterToCompiledCodeBridge directly for setEntryPointFromInterpreter but only setEntryPointFromJni
art/runtime/instrumentation.cc	

²²<https://www.jianshu.com/p/f45ce55c088c>

²³<https://www.jianshu.com/p/4310756f590e>

²⁴<https://www.dalvik.work/2021/05/28/dalvik-art-heap-gc/>

Procedure	Description
Instrumentation:: InitializeMethodsCode	
UpdateEntrypoints	Called multiple times for GetQuickToInterpreterBridge, GetQuickInstrumentationEntryPoint, GetQuickGenericJniStub, GetQuickToInterpreterBridge, GetQuickResolutionStub, interpreter:: GetNterpEntryPoint, GetQuickGenericJniStub, GetQuickInstrumentationEntryPoint, GetOptimizedCodeFor
art/runtime/entrypoints/runtime_asm_entrypoints.h GetQuickToInterpreterBridge	
art/runtime/arch/arm64/quick_entrypoints_arm64.S art_quick_to_ interpreter_bridge	
art/runtime/interpreter/interpreter.cc ArtInterpreterToInterpreterBridge	
art/runtime/arch/arm64/quick_entrypoints_arm64.S art_quick_resolution_ trampoline	
art/runtime/entrypoints/quick/quick_trampoline_entrypoints.cc artQuickResolutionTrampoline	
art/runtime/oat.cc OatHeader:: GetQuickResolutionTrampoline	Should be defined in the class_linker.cc, but AOSP13, it's in oat.cc. Supposed to return quick_resolution_trampoline_ is now defined in both oat_writer.h and class_linker.h.
art/runtime/native/dalvik_system_DexFile.cc DexFile_ defineClassNative	
art/runtime/class_linker.cc ClassLinker:: RegisterDexFileLocked ClassLinker:: RegisterDexFile	

Procedure	Description
ClassLinker:: AllocDexCache	art/runtime/mirror/dex_cache.cc
Dex:: RegisterDexFileLocked	art/runtime/class_linker-inl.h
ClassLinker:: ResolveMethod	art/runtime/class_linker.cc
ClassLinker:: DefineClass	art/runtime/class_linker.cc
ClassLinker:: LoadMethod	art/runtime/class_linker.cc

Table 6.7: ART Execution Process

Chapter 7

May

7.1 May 05-06, 2024

7.1.1 Run Zicheng's code without the weird text files

The logcat is still showing old logic even though I swapped files back. So I changed log output from "WEIMINN" to "WEIMINN2" but I got this error:

```
1 [ 8% 85/1028] //art/build/boot:art-bootclasspath-fragment dexpreopt art j
2 FAILED: out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib
3 /arm64/boot-apache-xml.art out/soong/cheetah/dex_artjars/android/apex/art_
4 boot_images/javalib/arm64/boot-apache-xml.oat out/soong/cheetah/dex_artjar
5 s/android/apex/art_boot_images/javalib/arm64/boot-apache-xml.vdex out/soon
6 g/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm64/boot-boun
7 cycastle.art out/soong/cheetah/dex_artjars/android/apex/art_boot_images/ja
8 valib/arm64/boot-bouncycastle.oat out/soong/cheetah/dex_artjars/android/ap
9 ex/art_boot_images/javalib/arm64/boot-bouncycastle.vdex out/soong/cheetah/
10 dex_artjars/android/apex/art_boot_images/javalib/arm64/boot-core-libart.ar
11 t out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm64
12 /boot-core-libart.oat out/soong/cheetah/dex_artjars/android/apex/art_boot_
13 images/javalib/arm64/boot-core-libart.vdex out/soong/cheetah/dex_artjars/a
14 ndroid/apex/art_boot_images/javalib/arm64/boot-okhttp.art out/soong/cheet
15 h/dex_artjars/android/apex/art_boot_images/javalib/arm64/boot-okhttp.oat o
16 ut/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm64/bo
17 ot-okhttp.vdex out/soong/cheetah/dex_artjars/android/apex/art_boot_images/
18 javalib/arm64/boot.art out/soong/cheetah/dex_artjars/android/apex/art_boot_
19 _images/javalib/arm64/boot.invocation out/soong/cheetah/dex_artjars/androi
20 d/apex/art_boot_images/javalib/arm64/boot.oat out/soong/cheetah/dex_artjar
21 s/android/apex/art_boot_images/javalib/arm64/boot.vdex out/soong/cheetah/d
22 ex_artjars_unstripped/android/apex/art_boot_images/javalib/arm64/boot-apac
23 he-xml.oat out/soong/cheetah/dex_artjars_unstripped/android/apex/art_boot_
24 images/javalib/arm64/boot-bouncycastle.oat out/soong/cheetah/dex_artjars_u
25 nstripped/android/apex/art_boot_images/javalib/arm64/boot-core-libart.oat
26 out/soong/cheetah/dex_artjars_unstripped/android/apex/art_boot_images/java
27 lib/arm64/boot-okhttp.oat out/soong/cheetah/dex_artjars_unstripped/android
28 /apex/art_boot_images/javalib/arm64/boot.oat
29 mkdir -p out/soong/cheetah/dex_artjars_unstripped/android/apex/art_boot_im
30 ages/javalib/arm64 && rm -f out/soong/cheetah/dex_artjars_unstripped/andro
31 id/apex/art_boot_images/javalib/arm64/*.art out/soong/cheetah/dex_artjars_
32 unstripped/android/apex/art_boot_images/javalib/arm64/*.oat out/soong/chee
```

```

33 tah/dex_artjars_unstripped/android/apex/art_boot_images/javalib/arm64/*.in
34 vocation && rm -f out/soong/cheetah/dex_artjars/android/apex/art_boot_imag
35 es/javalib/arm64/*.art out/soong/cheetah/dex_artjars/android/apex/art_boot
36 _images/javalib/arm64/*.oat out/soong/cheetah/dex_artjars/android/apex/art
37 _boot_images/javalib/arm64/*.invocation && ANDROID_LOG_TAGS="*:e" out/host
38 /linux-x86/bin/dex2oatd --avoid-storing-invocation --write-invocation-to=o
39 ut/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm64/bo
40 ot.invocation --runtime-arg -Xms64m --runtime-arg -Xmx64m --profile-file=o
41 ut/soong/cheetah/dex_artjars/boot.prof --dirty-image-objects=frameworks/ba
42 se/config/dirty-image-objects --base=0x70000000 --preloaded-classes=art/bu
43 ild/boot/preloaded-classes --dex-file=out/soong/cheetah/dex_artjars_input/
44 core-obj.jar --dex-file=out/soong/cheetah/dex_artjars_input/core-libart.jar
45 --dex-file=out/soong/cheetah/dex_artjars_input/okhttp.jar --dex-file=out/
46 soong/cheetah/dex_artjars_input/bouncycastle.jar --dex-file=out/soong/chee
47 tah/dex_artjars_input/apache-xml.jar --dex-location=/apex/com.android.art/
48 javalib/core-obj.jar --dex-location=/apex/com.android.art/javalib/core-liba
49 rt.jar --dex-location=/apex/com.android.art/javalib/okhttp.jar --dex-locat
50 ion=/apex/com.android.art/javalib/bouncycastle.jar --dex-location=/apex/co
51 m.android.art/javalib/apache-xml.jar --generate-debug-info --generate-buil
52 d-id --image-format=lz4hc --oat-symbols=out/soong/cheetah/dex_artjars_unst
53 ripped/android/apex/art_boot_images/javalib/arm64/boot.oat --strip --oat-f
54 ile=out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm
55 64/boot.oat --oat-location=out/soong/cheetah/dex_artjars/android/apex/art_
56 boot_images/javalib/boot.oat --image=out/soong/cheetah/dex_artjars/android
57 /apex/art_boot_images/javalib/arm64/boot.art --instruction-set=arm64 --and
58 roid-root=out/empty --no-inline-from=core-obj.jar --force-determinism --abo
59 rt-on-hard-verifier-error --instruction-set-variant=cortex-a55 --instructi
60 on-set-features=default --generate-mini-debug-info || ( echo 'ERROR: Dex2o
61 at failed to compile a boot image. It is likely that the boot classpath is
62 inconsistent. Rebuild with ART_BOOT_IMAGE_EXTRA_ARGS="--runtime-arg -verbos
63 e:verifier" to see verification errors.' ; false ) # hash of input list: b
64 b4c783e60c5fe8f723f06a1fc27caebedb88883dae99b303257caf0b2276d92
65 dex2oatd F 05-05 20:54:42 158 158 art_method.h:604] Check failed: IsIm
66 agePointerSize(pointer_size)
67 Runtime aborting...
68 All threads:
69 DALVIK THREADS (1):
70 "main" prio=5 tid=1 Runnable (still starting up)
71 | group="" sCount=0 ucsCount=0 flags=0 obj=(nil) self=0x64b1e64dbe30
72 | sysTid=158 nice=19 cgrp=default sched=0/0 handle=0x7dbfeaf10740
73 | state=R schedstat=( 23324439 37432 5 ) utm=1 stm=0 core=3 HZ=100
74 | stack=0x7ffd468d7000-0x7ffd468d9000 stackSize=8192KB
75 | held mutexes= "abort lock" "mutator lock"(shared held)
76 native: #00 pc 000000000e9a168 /home/weiminn/Documents/aosp13/out/host
77 /linux-x86/bin/dex2oatd64 (UnwindStackCurrent::UnwindFromContext(unsigned
78 long, void*)+88)
79 native: #01 pc 0000000000e94d9d /home/weiminn/Documents/aosp13/out/host
80 /linux-x86/bin/dex2oatd64 (art::DumpNativeStack(std::__1::basic_ostream<ch
81 ar, std::__1::char_traits<char> >&, int, BacktraceMap*, char const*, art::
82 ArtMethod*, void*, bool)+141)
83 native: #02 pc 0000000000ec8616 /home/weiminn/Documents/aosp13/out/host
84 /linux-x86/bin/dex2oatd64 (art::Thread::DumpStack(std::__1::basic_ostream<
85 char, std::__1::char_traits<char> >&, bool, BacktraceMap*, bool) const+422
86 )
87 native: #03 pc 0000000000ef19ff /home/weiminn/Documents/aosp13/out/host
88 /linux-x86/bin/dex2oatd64 (art::DumpCheckpoint::Run(art::Thread*)+575)
89 native: #04 pc 0000000000eea690 /home/weiminn/Documents/aosp13/out/host
90 /linux-x86/bin/dex2oatd64 (art::ThreadList::RunCheckpoint(art::Closure*, a
91 rt::Closure*)+288)

```

```
92      native: #05 pc 0000000000eea275  /home/weiminn/Documents/aosp13/out/host
93  /linux-x86/bin/dex2oatd64 (art::ThreadList::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> &, bool)+357)
94      native: #06 pc 0000000000dbae0a  /home/weiminn/Documents/aosp13/out/host
95  /linux-x86/bin/dex2oatd64 (art::AbortState::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> &)) const+186)
96      native: #07 pc 0000000000d9f46f  /home/weiminn/Documents/aosp13/out/host
97  /linux-x86/bin/dex2oatd64 (art::Runtime::Abort(char const*)+399)
98      native: #08 pc 000000000016cf01c  /home/weiminn/Documents/aosp13/out/host
99  /linux-x86/bin/dex2oatd64 (android::base::SetAborter(std::__1::function<void (char const*)>&&):$_3::__invoke(char const*)+60)
100     native: std::__1::__compressed_pair_elem<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>>::__rep, 0, false>::__compressed_pair_elem()
101     native: external/libcxx/include/memory:2140
102     native: std::__1::__compressed_pair<std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>>::__compressed_pair<true, void>()
103     native: external/libcxx/include/memory:2234
104     native: std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>>::basic_string<decltype(nullptr)>(char const*)
105     native: external/libcxx/include/string:820
106     native: UnwindStackCurrent::UnwindFromContext(unsigned long, void*)
107     native: system/unwinding/libbacktrace/UnwindStack.cpp:180
108     native: art::DumpNativeStack(std::__1::basic_ostream<char, std::__1::char_traits<char> &), int, BacktraceMap*, char const*, art::ArtMethod*, void*, bool)
109     native: art/runtime/native_stack_dump.cc:339
110     native: #09 pc 00000000016cf9a8  /home/weiminn/Documents/aosp13/out/host
111  /linux-x86/bin/dex2oatd64 (android::base::LogMessage::~LogMessage()+328)
112     native: art::Thread::DumpStack(std::__1::basic_ostream<char, std::__1::char_traits<char> &, bool, BacktraceMap*, bool) const
113     native: art/runtime/thread.cc:2298
114     native: #10 pc 0000000006000f2  /home/weiminn/Documents/aosp13/out/host
115  /linux-x86/bin/dex2oatd64 (art::ArtMethod::SetDataPtrSize(void const*, art::PointerSize)+226)
116     native: art::DumpCheckpoint::Run(art::Thread*)
117     native: art/runtime/thread_list.cc:213
118     native: art::ThreadList::RunCheckpoint(art::Closure*, art::Closure*)
119     native: art/runtime/thread_list.cc:374
120     native: art::ThreadList::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> &), bool)
121     native: art/runtime/thread_list.cc:257
122     native: art::AbortState::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> &)) const
123     native: art/runtime/runtime.cc:761
124     native: art::Dumpable<art::AbortState>::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> &)
125     native: art/libartbase/base/dumpable.h:38
126     native: std::__1::basic_ostream<char, std::__1::char_traits<char> &>) const
127     native: art/libartbase/base/dumpable.h:49
128     native: art::Runtime::Abort(char const*)
129     native: art/runtime/runtime.cc:884
130     native: #11 pc 0000000000daf16d  /home/weiminn/Documents/aosp13/out/host
131  /linux-x86/bin/dex2oatd64 (art::Runtime::CreateResolutionMethod()+285)
132     native: #12 pc 0000000000741453  /home/weiminn/Documents/aosp13/out/host
133     native: art::ClassLinker::InitWithoutImage(std::__1::vector<art::Image> &)
```

```

151  ctor<std::unique_ptr<art::DexFile const, std::default_delete<art
152  ::DexFile const> >, std::allocator<std::unique_ptr<art::DexFile
153  const, std::default_delete<art::DexFile const> > >, std::basic
154  _string<char, std::char_traits<char>, std::allocator<char> >*>+6
155  307)
156      native: #13 pc 0000000000da4709  /home/weiminn/Documents/aosp13/out/host
157  /linux-x86/bin/dex2oatd64 (art::Runtime::Init(art::RuntimeArgumentMap&&)+1
158  8073)
159      native: #14 pc 0000000000d9fef7  /home/weiminn/Documents/aosp13/out/host
160  /linux-x86/bin/dex2oatd64 (art::Runtime::Create(art::RuntimeArgumentMap&&)
161  +135)
162      native: #15 pc 0000000000491e0f  /home/weiminn/Documents/aosp13/out/host
163  /linux-x86/bin/dex2oatd64 (art::Dex2oat::CreateRuntime(art::RuntimeArgumentMap&&)
164  +127)
165      native: #16 pc 0000000000484526  /home/weiminn/Documents/aosp13/out/host
166  /linux-x86/bin/dex2oatd64 (art::Dex2oat::Setup()+2118)
167      native: #17 pc 000000000047edcc  /home/weiminn/Documents/aosp13/out/host
168  /linux-x86/bin/dex2oatd64 (main+1452)
169      native: android::base::SetAborter(std::function<void (char const*)
170  >&&):$_3::__invoke(char const*)
171      native: external/libcxx/include/functional:1799
172      native: android::base::LogMessage::~LogMessage()
173      native: system/libbase/logging.cpp:504
174      native: art::ArtMethod::SetDataPtrSize(void const*, art::PointerSize)
175      native: art/runtime/art_method.h:604
176      native: art::ArtMethod::SetEntryPointFromJniPtrSize(void const*, art::
177  PointerSize)
178      native: art/runtime/art_method.h:594
179      native: art::Runtime::CreateResolutionMethod()
180      native: art/runtime/runtime.cc:2789
181      native: art::ClassLinker::InitWithoutImage(std::vector<std::unique_ptr<art::DexFile const, std::default_delete<art::DexFile const> >, std::allocator<std::unique_ptr<art::DexFile const, std::default_delete<art::DexFile const> > >, std::basic_string<char, std::char_traits<char>, std::allocator<char> >*)
182      native: art/runtime/class_linker.cc:787
183      native: art::Runtime::Init(art::RuntimeArgumentMap&&)
184      native: art/runtime/runtime.cc:2077
185      native: art::Runtime::Create(art::RuntimeArgumentMap&&)
186      native: art/runtime/runtime.cc:1046
187      native: art::Dex2oat::CreateRuntime(art::RuntimeArgumentMap&&)
188      native: art/dex2oat/dex2oat.cc:2711
189      native: art::Dex2oat::Setup()
190      native: art/dex2oat/dex2oat.cc:1522
191      native: main
192      native: art/dex2oat/dex2oat.cc:3168
193      native: #18 pc 000000000002814f  /usr/lib/x86_64-linux-gnu/libc.so.6 (???
194  ?) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
195      native: #19 pc 0000000000028208  /usr/lib/x86_64-linux-gnu/libc.so.6 (__
196  libc_start_main+136) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
197      native: __libc_start_main_implementation
198      native: ./csu/../csu/libc-start.c:360
199      native: #20 pc 00000000004241a8  /home/weiminn/Documents/aosp13/out/host
200  /linux-x86/bin/dex2oatd64 (????)
201      (no managed stack frames)
202
203  dex2oatd E 05-05 20:54:51      158      158 thread-inl.h:170] holding "abort loc
204  k" at point where thread suspension is expected
205  Aborting thread:

```

```
210 "main" prio=5 tid=1 Runnable (still starting up)
211     | group="" sCount=0 ucsCount=0 flags=0 obj=(nil) self=0x64b1e64dbe30
212     | sysTid=158 nice=19 cgrp=default sched=0/0 handle=0x7dbfeaf10740
213     | state=R schedstat=( 46296127 29382693 40 ) utm=4 stm=0 core=8 HZ=100
214     | stack=0x7ffd468d7000-0x7ffd468d9000 stackSize=8192KB
215     | held mutexes= "abort lock" "mutator lock"(shared held)
216     native: #00 pc 0000000000e9a168 /home/weiminn/Documents/aosp13/out/host
217 /linux-x86/bin/dex2oatd64 (UnwindStackCurrent::UnwindFromContext(unsigned
218 long, void*)+88)
219     native: #01 pc 0000000000e94d9d /home/weiminn/Documents/aosp13/out/host
220 /linux-x86/bin/dex2oatd64 (art::DumpNativeStack(std::__1::basic_ostream<ch
221 ar, std::__1::char_traits<char> &, int, BacktraceMap*, char const*, art::
222 ArtMethod*, void*, bool)+141)
223     native: #02 pc 0000000000ec8616 /home/weiminn/Documents/aosp13/out/host
224 /linux-x86/bin/dex2oatd64 (art::Thread::DumpStack(std::__1::basic_ostream<
225 char, std::__1::char_traits<char> &, bool, BacktraceMap*, bool) const+422
226 )
227     native: #03 pc 0000000000dbb0dd /home/weiminn/Documents/aosp13/out/host
228 /linux-x86/bin/dex2oatd64 (art::AbortState::DumpThread(std::__1::basic_ost
229 ream<char, std::__1::char_traits<char> &, art::Thread*) const+93)
230     native: #04 pc 0000000000d9f46f /home/weiminn/Documents/aosp13/out/host
231 /linux-x86/bin/dex2oatd64 (art::Runtime::Abort(char const*)+399)
232     native: std::__1::__compressed_pair_elem<std::__1::basic_string<char,
233 std::__1::char_traits<char>, std::__1::allocator<char> >::__rep, 0, false>
234 ::__compressed_pair_elem()
235     native: external/libcxx/include/memory:2140
236     native: std::__1::__compressed_pair<std::__1::basic_string<char, std::__
237 __1::char_traits<char>, std::__1::allocator<char> >::__rep, std::__1::allo
238 cator<char> >::__compressed_pair<true, void>()
239     native: external/libcxx/include/memory:2234
240     native: std::__1::basic_string<char, std::__1::char_traits<char>, std::__
241 __1::allocator<char> >::basic_string<decltype(nullptr)>(char const*)
242     native: external/libcxx/include/string:820
243     native: UnwindStackCurrent::UnwindFromContext(unsigned long, void*)
244     native: system/unwinding/libbacktrace/UnwindStack.cpp:180
245     native: art::DumpNativeStack(std::__1::basic_ostream<char, std::__1::c
246 har_traits<char> &, int, BacktraceMap*, char const*, art::ArtMethod*, voi
247 d*, bool)
248     native: art/runtime/native_stack_dump.cc:339
249     native: #05 pc 00000000016cf1dc /home/weiminn/Documents/aosp13/out/host
250 /linux-x86/bin/dex2oatd64 (android::base::SetAborter(std::__1::function<vo
251 id (char const*)>&&)::$__3::__invoke(char const*)+60)
252     native: art::Thread::DumpStack(std::__1::basic_ostream<char, std::__1::c
253 har_traits<char> &, bool, BacktraceMap*, bool) const
254     native: art/runtime/thread.cc:2298
255     native: art::AbortState::DumpThread(std::__1::basic_ostream<char, std::__
256 __1::char_traits<char> &, art::Thread*) const
257     native: art/runtime/runtime.cc:782
258     native: art::Dumpable<art::AbortState>::Dump(std::__1::basic_ostream<c
259 har, std::__1::char_traits<char> &)
260     native: art/libartbase/base/dumpable.h:38
261     native: std::__1::basic_ostream<char, std::__1::char_traits<char> & a
262 rt::operator<< <art::AbortState>(std::__1::basic_ostream<char, std::__1::c
263 har_traits<char> &, art::Dumpable<art::AbortState> const&)
264     native: art/libartbase/base/dumpable.h:49
265     native: art::Runtime::Abort(char const*)
266     native: art/runtime/runtime.cc:884
267     native: #06 pc 00000000016cf9a8 /home/weiminn/Documents/aosp13/out/host
268 /linux-x86/bin/dex2oatd64 (android::base::LogMessage::~LogMessage() +328)
```

```

269      native: #07 pc 0000000006000f2  /home/weiminn/Documents/aosp13/out/host
270  /linux-x86/bin/dex2oatd64 (art::ArtMethod::SetDataPtrSize(void const*, art
271  ::PointerSize)+226)
272      native: #08 pc 000000000daf16d  /home/weiminn/Documents/aosp13/out/host
273  /linux-x86/bin/dex2oatd64 (art::Runtime::CreateResolutionMethod()+285)
274      native: #09 pc 0000000000741453  /home/weiminn/Documents/aosp13/out/host
275  /linux-x86/bin/dex2oatd64 (art::ClassLinker::InitWithoutImage(std::__1::ve
276  ctor<std::__1::unique_ptr<art::DexFile const, std::__1::default_delete<art
277  ::DexFile const> >, std::__1::allocator<std::__1::unique_ptr<art::DexFile
278  const, std::__1::default_delete<art::DexFile const> > >, std::__1::basic
279  _string<char, std::__1::char_traits<char>, std::__1::allocator<char> >)*6
280  307)
281      native: #10 pc 0000000000da4709  /home/weiminn/Documents/aosp13/out/host
282  /linux-x86/bin/dex2oatd64 (art::Runtime::Init(art::RuntimeArgumentMap&&)+1
283  8073)
284      native: android::base::SetAborter(std::__1::function<void (char const*
285  )>&&):$__3::__invoke(char const*)
286      native: external/libcxx/include/functional:1799
287      native: android::base::LogMessage::~LogMessage()
288      native: system/libbase/logging.cpp:504
289      native: art::ArtMethod::SetDataPtrSize(void const*, art::PointerSize)
290      native: art/runtime/art_method.h:604
291      native: art::ArtMethod::SetEntryPointFromJniPtrSize(void const*, art::
292  PointerSize)
293      native: art/runtime/art_method.h:594
294      native: art::Runtime::CreateResolutionMethod()
295      native: art/runtime/runtime.cc:2789
296      native: art::ClassLinker::InitWithoutImage(std::__1::vector<std::__1::
297  unique_ptr<art::DexFile const, std::__1::default_delete<art::DexFile const
298  > >, std::__1::allocator<std::__1::unique_ptr<art::DexFile const, std::__1
299  ::default_delete<art::DexFile const> > >, std::__1::basic_string<char, s
300  td::__1::char_traits<char>, std::__1::allocator<char> >)*
301      native: art/runtime/class_linker.cc:787
302      native: art::Runtime::Init(art::RuntimeArgumentMap&&)
303      native: art/runtime/runtime.cc:2077
304      native: #11 pc 0000000000d9fef7  /home/weiminn/Documents/aosp13/out/host
305  /linux-x86/bin/dex2oatd64 (art::Runtime::Create(art::RuntimeArgumentMap&&)
306  +135)
307      native: art::Runtime::Create(art::RuntimeArgumentMap&&)
308      native: art/runtime/runtime.cc:1046
309      native: #12 pc 0000000000491e0f  /home/weiminn/Documents/aosp13/out/host
310  /linux-x86/bin/dex2oatd64 (art::Dex2oat::CreateRuntime(art::RuntimeArgumen
311  tMap&&)+127)
312      native: art::Dex2oat::CreateRuntime(art::RuntimeArgumentMap&&)
313      native: art/dex2oat/dex2oat.cc:2711
314      native: #13 pc 0000000000484526  /home/weiminn/Documents/aosp13/out/host
315  /linux-x86/bin/dex2oatd64 (art::Dex2oat::Setup()+2118)
316      native: art::Dex2oat::Setup()
317      native: art/dex2oat/dex2oat.cc:1522
318      native: #14 pc 000000000047edcc  /home/weiminn/Documents/aosp13/out/host
319  /linux-x86/bin/dex2oatd64 (main+1452)
320      native: main
321      native: art/dex2oat/dex2oat.cc:3168
322      native: #15 pc 000000000002814f  /usr/lib/x86_64-linux-gnu/libc.so.6 (???
323  ?) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
324      native: #16 pc 0000000000028208  /usr/lib/x86_64-linux-gnu/libc.so.6 (__
325  libc_start_main+136) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
326      native: __libc_start_main_impl
327      native: ./csu/../csu/libc-start.c:360

```

```
328      native: #17 pc 0000000004241a8  /home/weiminn/Documents/aosp13/out/host
329 /linux-x86/bin/dex2oatd64 (???)  
330      (no managed stack frames)  
331 Aborted (core dumped)  
332 ERROR: Dex2oat failed to compile a boot image. It is likely that the boot c  
333 lasspath is inconsistent. Rebuild with ART_BOOT_IMAGE_EXTRA_ARGS="--runtime  
334 -arg -verbose:verifier" to see verification errors.  
335 [ 8% 86/1028] //art/build/boot:art-bootclasspath-fragment dexpreeopt art j  
336 FAILED: out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib  
337 /arm/boot-apache-xml.art out/soong/cheetah/dex_artjars/android/apex/art_bo  
338 ot_images/javalib/arm/boot-apache-xml.oat out/soong/cheetah/dex_artjars/an  
339 droid/apex/art_boot_images/javalib/arm/boot-apache-xml.vdex out/soong/chee  
340 tah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot-bouncycastle  
341 .art out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/ar  
342 m/boot-bouncycastle.oat out/soong/cheetah/dex_artjars/android/apex/art_boo  
343 t_images/javalib/arm/boot-bouncycastle.vdex out/soong/cheetah/dex_artjars/  
344 android/apex/art_boot_images/javalib/arm/boot-core-libart.art out/soong/ch  
345 eetah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot-core-libar  
346 t.oat out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/a  
347 rm/boot-core-libart.vdex out/soong/cheetah/dex_artjars/android/apex/art_boo  
348 t_images/javalib/arm/boot-okhttp.art out/soong/cheetah/dex_artjars/android/  
349 /apex/art_boot_images/javalib/arm/boot-okhttp.oat out/soong/cheetah/dex_a  
350 rtjars/android/apex/art_boot_images/javalib/arm/boot-okhttp.vdex out/soong  
351 /cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot.art out  
352 /soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot.i  
353 nvocation out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javal  
354 ib/arm/boot.oat out/soong/cheetah/dex_artjars/android/apex/art_boot_images  
355 /javalib/arm/boot.vdex out/soong/cheetah/dex_artjars_unstripped/android/ap  
356 ex/art_boot_images/javalib/arm/boot-apache-xml.oat out/soong/cheetah/dex_a  
357 rtjars_unstripped/android/apex/art_boot_images/javalib/arm/boot-bouncycastle  
358 .oat out/soong/cheetah/dex_artjars_unstripped/android/apex/art_boot_imag  
359 es/javalib/arm/boot-core-libart.oat out/soong/cheetah/dex_artjars_unstripp  
360 ed/android/apex/art_boot_images/javalib/arm/boot-okhttp.oat out/soong/chee  
361 tah/dex_artjars_unstripped/android/apex/art_boot_images/javalib/arm/boot.o  
362 at  
363 mkdir -p out/soong/cheetah/dex_artjars_unstripped/android/apex/art_boot_im  
364 ages/javalib/arm && rm -f out/soong/cheetah/dex_artjars_unstripped/android  
365 /apex/art_boot_images/javalib/arm/*.art out/soong/cheetah/dex_artjars_unst  
366 ripped/android/apex/art_boot_images/javalib/arm/*.oat out/soong/cheetah/de  
367 x_artjars_unstripped/android/apex/art_boot_images/javalib/arm/*.invocation  
368 && rm -f out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javal  
369 ib/arm/*.art out/soong/cheetah/dex_artjars/android/apex/art_boot_images/ja  
370 valib/arm/*.oat out/soong/cheetah/dex_artjars/android/apex/art_boot_images  
371 /javalib/arm/*.invocation && ANDROID_LOG_TAGS="*:e" out/host/linux-x86/bin  
372 /dex2oatd --avoid-storing-invocation --write-invocation-to=out/soong/cheet  
373 ah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot.invocation --  
374 runtime-arg -Xms64m --runtime-arg -Xmx64m --profile-file=out/soong/cheetah/  
375 /dex_artjars/boot.prof --dirty-image-objects=frameworks/base/config/dirty-  
376 image-objects --base=0x70000000 --preloaded-classes=art/build/boot/preload  
377 ed-classes --dex-file=out/soong/cheetah/dex_artjars_input/core-oj.jar --de  
378 x-file=out/soong/cheetah/dex_artjars_input/core-libart.jar --dex-file=out/  
379 soong/cheetah/dex_artjars_input/okhttp.jar --dex-file=out/soong/cheetah/de  
380 x_artjars_input/bouncycastle.jar --dex-file=out/soong/cheetah/dex_artjars_  
381 input/apache-xml.jar --dex-location=/apex/com.android.art/javalib/core-oj.  
382 jar --dex-location=/apex/com.android.art/javalib/core-libart.jar --dex-loc  
383 ation=/apex/com.android.art/javalib/okhttp.jar --dex-location=/apex/com.an  
384 droid.art/javalib/bouncycastle.jar --dex-location=/apex/com.android.art/ja  
385 valib/apache-xml.jar --generate-debug-info --generate-build-id --image-for  
386 mat=lz4hc --oat-symbols=out/soong/cheetah/dex_artjars_unstripped/android/a
```

```

387 pex/art_boot_images/javalib/arm/boot.oat --strip --oat-file=out/soong/chee
388 tah/dex_artjars/android/apex/art_boot_images/javalib/arm/boot.oat --oat-lo
389 cation=out/soong/cheetah/dex_artjars/android/apex/art_boot_images/javalib/
390 boot.oat --image=out/soong/cheetah/dex_artjars/android/apex/art_boot_image
391 s/javalib/arm/boot.art --instruction-set=arm --android-root=out/empty --no
392 -inline-from=core-obj.jar --force-determinism --abort-on-hard-verifier-erro
393 r --instruction-set-variant=generic --instruction-set-features=default --g
394 enerate-mini-debug-info || ( echo 'ERROR: Dex2oat failed to compile a boot
395     image. It is likely that the boot classpath is inconsistent. Rebuild with A
396 RT_BOOT_IMAGE_EXTRA_ARGS="--runtime-arg -verbose:verifier" to see verifica
397 tion errors.' ; false ) # hash of input list: bb4c783e60c5fe8f723f06a1fc27
398 caeedeb88883dae99b303257caf0b2276d92
399 dex2oatd F 05-05 20:54:42 159 159 art_method.h:604] Check failed: IsIm
400 agePointerSize(pointer_size)
401 Runtime aborting...
402 All threads:
403 DALVIK THREADS (1):
404 "main" prio=5 tid=1 Runnable (still starting up)
405 | group="" sCount=0 ucsCount=0 flags=0 obj=(nil) self=0x63f93b0df6f0
406 | sysTid=159 nice=19 cgrp=default sched=0/0 handle=0x7d8aae500740
407 | state=R schedstat=( 22175087 3260 6 ) utm=2 stm=0 core=10 HZ=100
408 | stack=0x7ffd9c378000-0x7ffd9c37a000 stackSize=8184KB
409 | held mutexes= "abort lock" "mutator lock"(shared held)
410     native: #00 pc 0000000000e9a168 /home/weiminn/Documents/aosp13/out/host
411 /linux-x86/bin/dex2oatd64 (UnwindStackCurrent::UnwindFromContext(unsigned
412 long, void*)+88)
413     native: #01 pc 0000000000e94d9d /home/weiminn/Documents/aosp13/out/host
414 /linux-x86/bin/dex2oatd64 (art::DumpNativeStack(std::__1::basic_ostream<ch
415 ar, std::__1::char_traits<char> >&, int, BacktraceMap*, char const*, art:
416 ArtMethod*, void*, bool)+141)
417     native: #02 pc 0000000000ec8616 /home/weiminn/Documents/aosp13/out/host
418 /linux-x86/bin/dex2oatd64 (art::Thread::DumpStack(std::__1::basic_ostream<
419 char, std::__1::char_traits<char> >&, bool, BacktraceMap*, bool) const+422
420 )
421     native: #03 pc 0000000000ef19ff /home/weiminn/Documents/aosp13/out/host
422 /linux-x86/bin/dex2oatd64 (art::DumpCheckpoint::Run(art::Thread*)+575)
423     native: #04 pc 0000000000eea690 /home/weiminn/Documents/aosp13/out/host
424 /linux-x86/bin/dex2oatd64 (art::ThreadList::RunCheckpoint(art::Closure*, a
425 rt::Closure*)+288)
426     native: #05 pc 0000000000eea275 /home/weiminn/Documents/aosp13/out/host
427 /linux-x86/bin/dex2oatd64 (art::ThreadList::Dump(std::__1::basic_ostream<c
428 har, std::__1::char_traits<char> >&, bool)+357)
429     native: #06 pc 0000000000dbae0a /home/weiminn/Documents/aosp13/out/host
430 /linux-x86/bin/dex2oatd64 (art::AbortState::Dump(std::__1::basic_ostream<c
431 har, std::__1::char_traits<char> >&) const+186)
432     native: #07 pc 0000000000d9f46f /home/weiminn/Documents/aosp13/out/host
433 /linux-x86/bin/dex2oatd64 (art::Runtime::Abort(char const*)+399)
434     native: #08 pc 00000000016cf1dc /home/weiminn/Documents/aosp13/out/host
435 /linux-x86/bin/dex2oatd64 (android::base::SetAborter(std::__1::function<vo
436 id (char const*)>&)::$::_3::__invoke(char const*)+60)
437     native: #09 pc 00000000016cf9a8 /home/weiminn/Documents/aosp13/out/host
438 /linux-x86/bin/dex2oatd64 (android::base::LogMessage::~LogMessage() +328)
439     native: #10 pc 00000000006000f2 /home/weiminn/Documents/aosp13/out/host
440 /linux-x86/bin/dex2oatd64 (art::ArtMethod::SetDataPtrSize(void const*, art
441 ::PointerSize)+226)
442     native: #11 pc 0000000000daf16d /home/weiminn/Documents/aosp13/out/host
443 /linux-x86/bin/dex2oatd64 (art::Runtime::CreateResolutionMethod() +285)
444     native: std::__1::__compressed_pair_elem<std::__1::basic_string<char,
445 std::__1::char_traits<char>, std::__1::allocator<char> >::__rep, 0, false>

```

```
446 ::__compressed_pair_elem()
447     native:    external/libcxx/include/memory:2140
448     native:    std::__1::__compressed_pair<std::__1::basic_string<char, std::
449 __1::char_traits<char>, std::__1::allocator<char> >::__rep, std::__1::allo
450 cator<char> >::__compressed_pair<true, void>()
451     native:    external/libcxx/include/memory:2234
452     native:    std::__1::basic_string<char, std::__1::char_traits<char>, std::
453 __1::allocator<char> >::__basic_string<decaytype(nullptr)>(char const*)
454     native:    external/libcxx/include/string:820
455     native:    UnwindStackCurrent::UnwindFromContext(unsigned long, void*)
456     native:    system/unwinding/libbacktrace/UnwindStack.cpp:180
457     native:    art::DumpNativeStack(std::__1::basic_ostream<char, std::__1::c
458 har_traits<char> >&, int, BacktraceMap*, char const*, art::ArtMethod*, voi
459 d*, bool)
460     native:    art/runtime/native_stack_dump.cc:339
461     native: #12 pc 0000000000741453 /home/weiminn/Documents/aosp13/out/host
462 /linux-x86/bin/dex2oatd64 (art::ClassLinker::InitWithoutImage(std::__1::ve
463 ctor<std::__1::unique_ptr<art::DexFile const, std::__1::default_delete<art
464 ::DexFile const> >, std::__1::allocator<std::__1::unique_ptr<art::DexFile
465 const, std::__1::default_delete<art::DexFile const> >>, std::__1::basic
466 _string<char, std::__1::char_traits<char>, std::__1::allocator<char> >)+6
467 307)
468     native:    art::Thread::DumpStack(std::__1::basic_ostream<char, std::__1:
469 :char_traits<char> >&, bool, BacktraceMap*, bool) const
470     native:    art/runtime/thread.cc:2298
471     native: #13 pc 0000000000da4709 /home/weiminn/Documents/aosp13/out/host
472 /linux-x86/bin/dex2oatd64 (art::Runtime::Init(art::RuntimeArgumentMap&&)+1
473 8073)
474     native:    art::DumpCheckpoint::Run(art::Thread*)
475     native:    art/runtime/thread_list.cc:213
476     native:    art::ThreadList::RunCheckpoint(art::Closure*, art::Closure*)
477     native:    art/runtime/thread_list.cc:374
478     native:    art::ThreadList::Dump(std::__1::basic_ostream<char, std::__1::c
479 har_traits<char> >&, bool)
480     native:    art/runtime/thread_list.cc:257
481     native:    art::AbortState::Dump(std::__1::basic_ostream<char, std::__1::c
482 har_traits<char> >&)
483     native:    art/runtime/runtime.cc:761
484     native:    art::Dumpable<art::AbortState>::Dump(std::__1::basic_ostream<c
485 har, std::__1::char_traits<char> >&)
486     native:    art/libartbase/base/dumpable.h:38
487     native:    std::__1::basic_ostream<char, std::__1::char_traits<char> >& a
488 rt::operator<< <art::AbortState>(std::__1::basic_ostream<char, std::__1::c
489 har_traits<char> >&, art::Dumpable<art::AbortState> const&)
490     native:    art/libartbase/base/dumpable.h:49
491     native:    art::Runtime::Abort(char const*)
492     native:    art/runtime/runtime.cc:884
493     native: #14 pc 0000000000d9fef7 /home/weiminn/Documents/aosp13/out/host
494 /linux-x86/bin/dex2oatd64 (art::Runtime::Create(art::RuntimeArgumentMap&&)
495 +135)
496     native: #15 pc 0000000000491e0f /home/weiminn/Documents/aosp13/out/host
497 /linux-x86/bin/dex2oatd64 (art::Dex2oat::CreateRuntime(art::RuntimeArgumen
498 tMap&&)+127)
499     native: #16 pc 0000000000484526 /home/weiminn/Documents/aosp13/out/host
500 /linux-x86/bin/dex2oatd64 (art::Dex2oat::Setup()+2118)
501     native: #17 pc 000000000047edcc /home/weiminn/Documents/aosp13/out/host
502 /linux-x86/bin/dex2oatd64 (main+1452)
503     native: #18 pc 00000000002814f /usr/lib/x86_64-linux-gnu/libc.so.6 (???
504 ?) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
```

```

505      native: #19 pc 0000000000028208  /usr/lib/x86_64-linux-gnu/libc.so.6 (__
506      libc_start_main+136) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
507      native:  __libc_start_main_implementation
508      native:  ./csu/../csu/libc-start.c:360
509      native: #20 pc 00000000004241a8  /home/weiminn/Documents/aosp13/out/host
510  /linux-x86/bin/dex2oatd64 (???)  

511      (no managed stack frames)
512
513 dex2oatd E 05-05 20:54:51  159  159 thread-inl.h:170] holding "abort loc
514 k" at point where thread suspension is expected
515 Aborting thread:
516 "main" prio=5 tid=1 Runnable (still starting up)
517   | group="" sCount=0 ucsCount=0 flags=0 obj=(nil) self=0x63f93b0df6f0
518   | sysTid=159 nice=19 cgrp=default sched=0/0 handle=0x7d8aae500740
519   | state=R schedstat=( 46410502 17203600 40 ) utm=4 stm=0 core=18 HZ=100
520   | stack=0x7ffd8c378000-0x7ffd8c37a000 stackSize=8184KB
521   | held mutexes= "abort lock" "mutator lock"(shared held)
522   native: #00 pc 0000000000e9a168  /home/weiminn/Documents/aosp13/out/host
523  /linux-x86/bin/dex2oatd64 (UnwindStackCurrent::UnwindFromContext(unsigned
524 long, void*)+88)
525   native: #01 pc 0000000000e94d9d  /home/weiminn/Documents/aosp13/out/host
526  /linux-x86/bin/dex2oatd64 (art::DumpNativeStack(std::__1::basic_ostream<ch
527 ar, std::__1::char_traits<char> >&, int, BacktraceMap*, char const*, art::
528 ArtMethod*, void*, bool)+141)
529   native: #02 pc 0000000000ec8616  /home/weiminn/Documents/aosp13/out/host
530  /linux-x86/bin/dex2oatd64 (art::Thread::DumpStack(std::__1::basic_ostream<
531 char, std::__1::char_traits<char> >&, bool, BacktraceMap*, bool) const+422
532 )
533   native: #03 pc 0000000000dbb0dd  /home/weiminn/Documents/aosp13/out/host
534  /linux-x86/bin/dex2oatd64 (art::AbortState::DumpThread(std::__1::basic_ost
535 ream<char, std::__1::char_traits<char> >&, art::Thread*) const+93)
536   native: #04 pc 0000000000d9f46f  /home/weiminn/Documents/aosp13/out/host
537  /linux-x86/bin/dex2oatd64 (art::Runtime::Abort(char const*)+399)
538   native:  std::__1::__compressed_pair_elem<std::__1::basic_string<char,
539 std::__1::char_traits<char>, std::__1::allocator<char> >::__rep, 0, false>
540  ::__compressed_pair_elem()
541   native:  external/libcxx/include/memory:2140
542   native:  std::__1::__compressed_pair<std::__1::basic_string<char, std::__
543 1::char_traits<char>, std::__1::allocator<char> >::__rep, std::__1::allo
544  cator<char> >::__compressed_pair<true, void>()
545   native:  external/libcxx/include/memory:2234
546   native:  std::__1::basic_string<char, std::__1::char_traits<char>, std::__
547 1::allocator<char> >::__basic_string<decltype(nullptr)>(char const*)
548   native:  external/libcxx/include/string:820
549   native:  UnwindStackCurrent::UnwindFromContext(unsigned long, void*)
550   native:  system/unwinding/libbacktrace/UnwindStack.cpp:180
551   native:  art::DumpNativeStack(std::__1::basic_ostream<char, std::__1::c
552 har_traits<char> >&, int, BacktraceMap*, char const*, art::ArtMethod*, voi
553 d*, bool)
554   native:  art/runtime/native_stack_dump.cc:339
555   native: #05 pc 00000000016cf01c  /home/weiminn/Documents/aosp13/out/host
556  /linux-x86/bin/dex2oatd64 (android::base::SetAborter(std::__1::function<vo
557 id (char const*)>&&):$__3::__invoke(char const*)+60)
558   native:  art::Thread::DumpStack(std::__1::basic_ostream<char, std::__1:
559  char_traits<char> >&, bool, BacktraceMap*, bool) const
560   native:  art/runtime/thread.cc:2298
561   native:  art::AbortState::DumpThread(std::__1::basic_ostream<char, std::__
562 1::char_traits<char> >&, art::Thread*) const
563   native:  art/runtime/runtime.cc:782

```

```
564     native:    art::Dumpable<art::AbortState>::Dump(std::__1::basic_ostream<c
565 har, std::__1::char_traits<char> &) const
566     native:    art/libartbase/base/dumpable.h:38
567     native:    std::__1::basic_ostream<char, std::__1::char_traits<char> &> a
568 rt::operator<< <art::AbortState>(std::__1::basic_ostream<char, std::__1::c
569 har_traits<char> &, art::Dumpable<art::AbortState> const&)
570     native:    art/libartbase/base/dumpable.h:49
571     native:    art::Runtime::Abort(char const*)
572     native:    art/runtime/runtime.cc:884
573     native: #06 pc 00000000016cf9a8  /home/weiminn/Documents/aosp13/out/host
574 /linux-x86/bin/dex2oatd64 (android::base::LogMessage::~LogMessage() +328)
575     native: #07 pc 00000000006000f2  /home/weiminn/Documents/aosp13/out/host
576 /linux-x86/bin/dex2oatd64 (art::ArtMethod::SetDataPtrSize(void const*, art
577 ::PointerSize)+226)
578     native: #08 pc 0000000000daf16d  /home/weiminn/Documents/aosp13/out/host
579 /linux-x86/bin/dex2oatd64 (art::Runtime::CreateResolutionMethod() +285)
580     native: #09 pc 000000000741453  /home/weiminn/Documents/aosp13/out/host
581 /linux-x86/bin/dex2oatd64 (art::ClassLinker::InitWithoutImage(std::__1::ve
582 ctor<std::__1::unique_ptr<art::DexFile const, std::__1::default_delete<art
583 ::DexFile const> >, std::__1::allocator<std::__1::unique_ptr<art::DexFile
584 const, std::__1::default_delete<art::DexFile const> >>, std::__1::basic
585 _string<char, std::__1::char_traits<char>, std::__1::allocator<char> >*) +6
586 307)
587     native: #10 pc 0000000000da4709  /home/weiminn/Documents/aosp13/out/host
588 /linux-x86/bin/dex2oatd64 (art::Runtime::Init(art::RuntimeArgumentMap&&) +1
589 8073)
590     native:    android::base::SetAborter(std::__1::function<void (char const*
591 )>&&):$._3::__invoke(char const*)
592     native:    external/libcxx/include/functional:1799
593     native:    android::base::LogMessage::~LogMessage()
594     native:    system/libbase/logging.cpp:504
595     native:    art::ArtMethod::SetDataPtrSize(void const*, art::PointerSize)
596     native:    art/runtime/art_method.h:604
597     native:    art::ArtMethod::SetEntryPointFromJniPtrSize(void const*, art::
598 PointerSize)
599     native:    art/runtime/art_method.h:594
600     native:    art::Runtime::CreateResolutionMethod()
601     native:    art/runtime/runtime.cc:2789
602     native:    art::ClassLinker::InitWithoutImage(std::__1::vector<std::__1::
603 unique_ptr<art::DexFile const, std::__1::default_delete<art::DexFile const
604 > >, std::__1::allocator<std::__1::unique_ptr<art::DexFile const, std::__1
605 ::default_delete<art::DexFile const> >>, std::__1::basic_string<char, std
606 ::__1::char_traits<char>, std::__1::allocator<char> >*)
607     native:    art/runtime/class_linker.cc:787
608     native:    art::Runtime::Init(art::RuntimeArgumentMap&&)
609     native:    art/runtime/runtime.cc:2077
610     native: #11 pc 0000000000d9fef7  /home/weiminn/Documents/aosp13/out/host
611 /linux-x86/bin/dex2oatd64 (art::Runtime::Create(art::RuntimeArgumentMap&&) +135)
612
613     native:    art::Runtime::Create(art::RuntimeArgumentMap&&)
614     native:    art/runtime/runtime.cc:1046
615     native: #12 pc 0000000000491e0f  /home/weiminn/Documents/aosp13/out/host
616 /linux-x86/bin/dex2oatd64 (art::Dex2oat::CreateRuntime(art::RuntimeArgument
617 Map&&) +127)
618     native:    art::Dex2oat::CreateRuntime(art::RuntimeArgumentMap&&)
619     native:    art/dex2oat/dex2oat.cc:2711
620     native: #13 pc 000000000484526  /home/weiminn/Documents/aosp13/out/host
621 /linux-x86/bin/dex2oatd64 (art::Dex2oat::Setup() +2118)
622     native:    art::Dex2oat::Setup()
```

```

623      native:  art/dex2oat/dex2oat.cc:1522
624      native: #14 pc 000000000047edcc  /home/weiminn/Documents/aosp13/out/host
625 /linux-x86/bin/dex2oatd64 (main+1452)
626      native:  main
627      native:  art/dex2oat/dex2oat.cc:3168
628      native: #15 pc 000000000002814f  /usr/lib/x86_64-linux-gnu/libc.so.6 (???
629 ?) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
630      native: #16 pc 0000000000028208  /usr/lib/x86_64-linux-gnu/libc.so.6 (__
631 libc_start_main+136) (BuildId: b20cbdb62d7717c13dc61a48b7b2e673a7edf233)
632      native:  __libc_start_main_Impl
633      native:  ./csu/../csu/libc-start.c:360
634      native: #17 pc 00000000004241a8  /home/weiminn/Documents/aosp13/out/host
635 /linux-x86/bin/dex2oatd64 (????)
636      (no managed stack frames)
637 Aborted (core dumped)
638 ERROR: Dex2oat failed to compile a boot image. It is likely that the boot c
639 lasspath is inconsistent. Rebuild with ART_BOOT_IMAGE_EXTRA_ARGS="--runtime
640 -arg -verbose:verifier" to see verification errors.
641 20:54:57 ninja failed with: exit status 1
642
643 ##### failed to build some targets (18 seconds) #####

```

Deleted the contents of the AOSP 13 project and resynced, and now got new:

```

1 [ 0% 107/19930] //packages/modules/NeuralNetworks/runtime:libneuralnetwor
2 FAILED: out/soong/.intermediates/packages/modules/NeuralNetworks/runtime/l
3 ibneuralnetworks/android_arm64_armv8-2a_cortex-a55_shared_apex30/libneural
4 networks.so.lsdump
5 prebuilt/clang-tools/linux-x86/bin/header-abi-linker --root-dir . --root-
6 dir $OUT_DIR:out -o out/soong/.intermediates/packages/modules/NeuralNetwor
7 ks/runtime/libneuralnetworks/android_arm64_armv8-2a_cortex-a55_shared_apex
8 30/libneuralnetworks.so.lsdump -so out/soong/.intermediates/packages/modul
9 es/NeuralNetworks/runtime/libneuralnetworks/android_arm64_armv8-2a_cortex-
10 a55_shared_apex30/libneuralnetworks.so -v packages/modules/NeuralNetworks/
11 runtime/libneuralnetworks.map.txt -arch arm64 -Ipackages/modules/NeuralNet
12 works/runtime/include -Iframeworks/nativecmds/lshal/libprocpartition/incl
13 ude -Ibuild/soong/cc/libbuildversion/include -Iout/soong/.intermediates/pa
14 ckages/modules/NeuralNetworks/runtime/statslog_neuralnetworks.h/gen @out/s
15 oong/.intermediates/packages/modules/NeuralNetworks/runtime/libneuralnetwo
16 rks/android_arm64_armv8-2a_cortex-a55_shared_apex30/libneuralnetworks.so.l
17 sdump.rsp
18 Failed to parse JSON: * Line 1, Column 1
19     Syntax error: value, object or array expected.
20
21 ReadDump failed
22 libc++abi: Pure virtual function called!
23 Aborted (core dumped)
24 21:57:03 ninja failed with: exit status 1

```

Also finding a lot of warning messages like:

```

1 no declaration found for ELF symbol with id acoshl@@LIBC
2 no declaration found for ELF symbol with id acosl@@LIBC
3 no declaration found for ELF symbol with id asinhl@@LIBC
4 no declaration found for ELF symbol with id asinl@@LIBC
5 no declaration found for ELF symbol with id atan2l@@LIBC
6 no declaration found for ELF symbol with id atanhl@@LIBC
7 no declaration found for ELF symbol with id atanl@@LIBC
8 no declaration found for ELF symbol with id cbrtl@@LIBC
9 no declaration found for ELF symbol with id ceill@@LIBC

```

```

10 no declaration found for ELF symbol with id cosh1@@LIBC
11 no declaration found for ELF symbol with id cosl@@LIBC
12 no declaration found for ELF symbol with id csqtl@@LIBC
13 no declaration found for ELF symbol with id erfcl@@LIBC
14 no declaration found for ELF symbol with id erfl@@LIBC
15 no declaration found for ELF symbol with id expm1@@LIBC
16 no declaration found for ELF symbol with id floor@@LIBC
17 no declaration found for ELF symbol with id floorl@@LIBC
18 no declaration found for ELF symbol with id fmal@@LIBC
19 no declaration found for ELF symbol with id frexp1@@LIBC
20 no declaration found for ELF symbol with id hypotl@@LIBC
21 no declaration found for ELF symbol with id ldexpf@@LIBC
22 no declaration found for ELF symbol with id ldexpl@@LIBC
23 no declaration found for ELF symbol with id lgammal@@LIBC
24 no declaration found for ELF symbol with id lgammal_r@@LIBC
25 no declaration found for ELF symbol with id log10l@@LIBC
26 no declaration found for ELF symbol with id log1pl@@LIBC
27 no declaration found for ELF symbol with id logbl@@LIBC
28 no declaration found for ELF symbol with id nanl@@LIBC
29 no declaration found for ELF symbol with id nextafterl@@LIBC
30 no declaration found for ELF symbol with id nexttoward@@LIBC
31 no declaration found for ELF symbol with id nexttowardl@@LIBC
32 no declaration found for ELF symbol with id remainderl@@LIBC
33 no declaration found for ELF symbol with id remquo1@@LIBC
34 no declaration found for ELF symbol with id rint1@@LIBC
35 no declaration found for ELF symbol with id scalbnl@@LIBC
36 no declaration found for ELF symbol with id sinh1@@LIBC
37 no declaration found for ELF symbol with id sinl@@LIBC
38 no declaration found for ELF symbol with id sqrt@@LIBC
39 no declaration found for ELF symbol with id sqrtf@@LIBC
40 no declaration found for ELF symbol with id sqrtl@@LIBC
41 no declaration found for ELF symbol with id tanh1@@LIBC
42 no declaration found for ELF symbol with id tanl@@LIBC
43 no declaration found for ELF symbol with id trunc1@@LIBC

```

Now it compiles successfully, but stuck in infinite loop at boot.

Check the driver number for Pixel 7 Pro¹ and the number is TD1A.220804.031. Now download the driver binary for TD1A.220804.031². And now it works! Turns out I forgot to run the driver install script after I synced the repo.

The logic is still not updating after I added in some logging code with [WEIMINN] tag inside the frameworks/base/core/jni/AndroidRuntime.cpp. The build system doesn't seem to be recognizing the changes in the source code. So I find the compiled framework objects in out using find./out-inameframework and deleted the entire out/soong/.intermediates/frameworks, and run the make command again. Now it recompiles, but still not updating the logic inside the flashed ROM. Rebooting the machine also doesn't work.

For the record, this is the command to find a string in the project:

```

1 clear && find ./frameworks/ ./art/ ./bionic/ ./device/ ./build/ ./libnativehelper
  \(-iname '*.java' -o -iname '*.cpp' -o -iname '*.cc' -o -iname '*.hpp' -o -
  -iname '*.h' -o -iname '*.S' -o -iname '*.mk' \) -a ! -iname '*test*' -a ! -

```

¹<https://source.android.com/docs/setup/reference/build-numbers>

²<https://developers.google.com/android/drivers>

```
iname '*out*' | xargs grep --color -sin 'docall('
```

Relaunch for emulator with this script:

```

1 echo "[WEIMINN] Building AOSP"
2 cd /home/weiminn/Documents/aosp13_new
3 source build/envsetup.sh
4 lunch sdk_phone_x86_64-userdebug
5 make
6
7 emulator &
8
9 boot_start='date +%s'
10 adb wait-for-device
11
12 echo "[WEIMINN] Waiting for boot completion"
13 A=$(adb shell getprop sys.boot_completed | tr -d '\r')
14 while [ "$A" != "1" ]; do
15     echo "[WEIMINN] Waiting for boot completion"
16     sleep .5
17     A=$(adb shell getprop sys.boot_completed | tr -d '\r')
18 done
19 boot_end='date +%s'
20 echo Boot time was `expr $boot_end - $boot_start` seconds.
21
22 adb root
23
24 echo "[WEIMINN] Enabling logging"
25 adb shell setprop debug.ld.all dlerror,dlopen
26
27 rm -f logcat.txt
28 echo "[WEIMINN] Logging with Logcat to logcat.txt"
29 adb logcat >> logcat.txt

```

And now the code changes are detected by the build system and the new logic is reflected in the logcat!

7.1.2 Implement Zicheng Code inside AOSP13 Emulator

Insert the following code at the bottom of build/make/target/board/emulator_x86_64/device.mk:

```

1 DISABLE_ARTIFACT_PATH_REQUIREMENTS := true
2 PRODUCT_PACKAGES += MINIMAManager

```

And paste in the MINIMAManager.apk inside packages/apps/MINIMAManager directory.

But the app installed via adbinstallsuccess_apps/60_com.atomic.apps.medical.drug.dictionary.apk doesn't launch after preloading the MINIMAManager APK. If I remove the preload, then the app launches.

Pasted in the Customized Framework codes and now can successfully adb install and launch the apps. The debloating also works!

7.2 May 07-14, 2024

7.2.1 Verify AOSP13 Debloating via ART Only

Get a list of APKs that can be successfully debloated and demoed.

```

1 adb install success_apps/60_com.atomic.apps.medical.drug.dictionary.apk
2 adb push success_schema/60_com.atomic.apps.medical.drug.dictionary_removed_methods
   .txt /sdcard
3
4 adb install success_apps/169_com.storiestime.apk
5 adb push success_schema/169_com.storiestime_removed_methods.txt /sdcard
6
7 adb install success_apps/471_com.mobilemirror.app2018.mirror.apk
8 adb push success_schema/471_com.mobilemirror.app2018.mirror_removed_methods.txt /
   sdcard
9
10 adb install success_apps/400_com.myprograms.glasgow.apk
11
12 adb install success_apps/382_com.diogines.pregnancy.apk

```

These APKs cannot be launched

```

1 adb install success_apps/363_com.BloodpressureHistoryDiary.healthRecordsAnalyze.
   apk
2 adb install success_apps/424_com.mdhelper.cardiojournal.apk

```

7.2.2 Which components are involved in Framework for Debloating?

Procedure	Description
services/core/java/com/android/server/pm/permission/PermissionManagerService.java onPackageInstalled	Modify to add READ Permission to the installed package's requested permissions.

Table 7.1: Framework Procedures

7.3 Isolate functionalities of ART Procedures

Procedure	Zicheng's Work
libartbase/base/os_linux.h	
CreateMINIMAFfile	Not invoked
CreateDirectories2	Not invoked
libartbase/base/os_linux.cc	
CreateMINIMAFfile	Not invoked
CreateDirectories2	Not invoked
libdexfile/dex/art_dex_file_loader.cc	

Procedure	Zicheng's Work
ArtDexFileLoader::Open	Only trivial logging
libdexfile/dex/dex_file_loader.cc	
DexFileLoader::OpenCommon	Only trivial logging
libnativeloader/native_loader_namespace.cpp	
NativeLoaderNamespace::Load	Only trivial logging
libnativeloader/native_loader.cpp	
OpenNativeLibrary	Only trivial logging
OpenNativeLibraryInNamespace	Only trivial logging
oderefresh/oderefresh.cc	
AddDex2oatProfileAndCompilerFilter	Change property dalvik.vm.systemservercompilerfilter for speed to verify.
oderefresh/oderefresh_main.cc	
InitializeConfig	Change argument --compiler-filter for speed to verify.
runtime/entrypoints/quick/quick_trampoline_entry_points.cc	
artQuickToInterpreterBridge	No change
runtime/interpreter/interpreter_common.cc	
DoCallCommon	Only trivial Logging
runtime/interpreter/interpreter_switch_impl-inl.h	
Additional header Macros	#include"utils/Log.h" #include<jni.h> #include"runtime_globals.h" #include"jni/java_vm_ext.h" #include"jni/jni_env_ext.h"
ExecuteSwitchImplCpp	Get SwitchImplContext's shadow_frame's GetMethod's PrettyMethod and check MYmatch_hook_method. If the method's signature matches, create intent to switch to MINIMA app.
runtime/interpreter/interpreter_switch_impl.h	
ExecuteSwitchImpl	No change
runtime/interpreter/interpreter.cc	
Execute	No change
EnterInterpreterFromEntryPoint	No change
runtime/jit/jit.cc	

Procedure	Zicheng's Work
MethodEntered	No change
runtime/jni/java_vm_ext.cc	
Additional header Macros	#include"utils/Log.h"
FindSymbol	Compare symbol_name to Java_edu_smu_imitatejumping_Simple_1JNI_1Class_duration
JavaVMExt::LoadNativeLibrary	Only trivial logging
runtime/jni/jni_internal.cc	
FindClass	No change
runtime/mirror/class.cc	
FindClassMethodWithSignature	No change
runtime/native/dalvik_system_DexFile.cc	
DexFile_openDexFileNative	No change
runtime/native/dalvik_system_ZygoteHooks.cc	
modifyFilePermissions1	Procedure to modify a given file's permission; Not invoked
ZygoteHooks_	Remove tracing logic. Get the process' package name for trivial logging.
nativePostForkChild	
runtime/app_info.cc	
GetPackageName (NO NEED)	Get package name from AppInfo instance
runtime/app_info.h	
GetPackageName (NO NEED)	Add definition to retrieve package name of the app; Not invoked
runtime/art_method-inl.h	
ArtMethod::	Trivial Logging
CheckIncompatibleClassChange	
ArtMethod::UpdateCounter	Get PrettyMethod and log it with hotness_count_
runtime/art_method.cc	
Additional headers	#include"utils/Log.h"
ArtMethod::Invoke (NO NEED)	Log its PrettyMethod if the method name matches Runtime::MYmatch_target_method after having quick code.
ArtMethod::PrettyMethod (NO NEED)	Add additional information such as DEX method index, counter, and method index
runtime/art_method.h	

Procedure	Zicheng's Work
ArtMethod::UpdateCounter	Modify declaration to include <code>REQUIRES_SHARED(Locks::mutator_lock_)</code> at the end
<code>runtime/class_linker.cc</code>	
<code>ClassLinker::FindClass</code>	Only trivial logging
<code>LinkCode</code>	<code>SetDontCompile</code> for the method if <code>Runtime::MYmatch_target_method</code> or <code>Runtime::MYmatch_hook_method</code> , plus trivial logging
<code>ClassLinker::LoadClass</code>	Only trivial Logging
<code>runtime/instrumentation.cc</code>	
<code>CanUseNterp</code>	If <code>Runtime::MYmatch_target_method</code> or <code>Runtime::MYmatch_hook_method</code> then return false to exit instead of proceeding.
<code>Instrumentation::InitializeMethodsCode</code>	Check for <code>empty_Hook()</code> in the method signature and then logs. Add logging during the check if method is native.
<code>runtime/oat_file_manager.cc</code>	
<code>OatFileManager::OpenDexFilesFromOat_Impl</code>	Only trivial logging
<code>runtime/runtime_globals.h</code>	
Added headers	<pre>#include"utils/Log.h" #include<iostream> #include<sstream> #include<fstream> #include<stdio.h> #include<dirent.h> #include"base/os.h" #include<vector> #include<string></pre>
<code>runtime/runtime.h</code>	
<code>NativeLibFunc</code>	Add declaration
<code>MYmatch_hook_method</code>	Add declaration
<code>MYmatch_target_method</code>	Add declaration
<code>MYreadFile</code>	Add declaration
<code>MYmake_file</code>	Add declaration
<code>read_content</code>	Add declaration
<code>getLoadingSchema</code>	Add definition
<code>getLoadedSchema</code>	Add definition

Procedure	Zicheng's Work
getShouldReadSchema	Add definition
setLoadingSchema	Add definition
setLoadedSchema	Add definition
setShouldReadSchema	Add definition; not invoked
getDebugging	Add definition
getTarget_method_string_vector	Add definition for accessor to target_method_string_vector
setTarget_method_string_vector	Add definition
getMINIMA_file_path	Add definition for accessor to MINIMA_file_path
getMINIMA_folder_path	Not invoked
getHookMethodName	Add definition for accessor to hook_method_name
MINIMA_file_path	Add private declaration
hook_method_name	Add private declaration
target_method_string_vector	Add private declaration
target_native_func_string_vector	Add private declaration
debugging	Add private declaration
api_prefix_List	Add private declaration
loaded_schema_	Add private declaration after image_dex2oat_enabled_
loading_schema_	Add private declaration after image_dex2oat_enabled_
should_read_schema_	Add private declaration after image_dex2oat_enabled_
runtime/runtime.cc	
StringAppendV	Called by StringPrintf and StringAppendF which are not invoked
StringPrintf	Not invoked
StringAppendF	Not invoked
Runtime::MINIMA_file_path	Hardcoded to "/data/local/tmp/mySchema.txt"
Runtime::hook_method_name	Hardcoded to "voidedu.smu.minimaconfig.SettingsActivity.empty_Hook()"
Runtime::debugging	Hardcoded to false
Runtime::target_method_string_vector	The vector that stores the schema
Runtime::target_native_func_string_vector	Empty schema of native functions. Not invoked
api_prefix_List	Not invoked

Procedure	Zicheng's Work
NativeLibFunc	Format to store native functions in schema target_native_func_string_vector
Runtime	Add initializers loaded_schema_(false), loading_schema_(false) and should_read_schema_(false).
Create	Checks if MINIMA file exists with OS::FileExists(Runtime::getMINIMA_file_path().c_str()) and does nothing with it
Start	Only trivial logging
Init	Remove trace_config_->trace_mode=Trace::TraceMode::kMethodTracing;
MYmatch_hook_method	Check if method is hook_method_name
java2cpp	Converts Java arraylist to C++ vector
is_within_JNImethod_list	Not invoked
Runtime::read_content	Loads the schema and stores it in Runtime::target_native_func_string_vector.
is_packageMethod	Not invoked
Runtime::MYmatch_target_method	Returns false if still loading schema (getLoadingSchema), runtime not started or is system server.
Runtime::MYmake_file	If preinstalled pacakge, should not read schema. Otherwise, call read_content to load the schema.
Runtime::MYreadFile	Access the schema via getTarget_method_string_vector and iterate through to see if the method matches any of the signature in the schema.
runtime/trace.cc	Not invoked
modifyFilePermissions	Not invoked
Trace::Start	Modify permission of given file
Trace::Trace	Calls modifyFilePermissions to modify trace_filename
Trace::MethodEntered	Change initialization of trace_mode_ from trace_mode to to TraceMode::kMethodTracing
Trace::MethodExited	No change

Table 7.2: ART Procedures involved in Debloating

In frameworks/base/services/core/java/com/android/server/pm/parsing/pkg/

PackageImpl.java, comment out this line in method void makeImmutable:

```
1 requestedPermissions = Collections.unmodifiableList(requestedPermissions);
```

7.4 May 15, 2024

7.4.1 Verify debloater in several android apps

```
1 05-15 21:44:13.512      0      0 E SELinux : avc: denied { find } for pid=2247 uid
      =99007 name=content_capture scontext=u:r:isolated_app:s0:c512,c768 tcontext=u:
      object_r:content_capture_service:s0 tclass=service_manager permissive=0
2 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517] No pending
      exception expected: java.lang.SecurityException: Isolated process not allowed
      to call getContentProvider
3 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at java.lang.
      Exception android.os.Parcel.createExceptionOrNull(int, java.lang.String) (
      Parcel.java:3057)
4 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at java.lang.
      Exception android.os.Parcel.createException(int, java.lang.String) (Parcel.
      java:3041)
5 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at void
      android.os.Parcel.readException(int, java.lang.String) (Parcel.java:3024)
6 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at void
      android.os.Parcel.readException() (Parcel.java:2966)
7 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.app
      .ContentProviderHolder android.app.IActivityManager$Stub$Proxy.
      getContentProvider(android.app.IApplicationThread, java.lang.String, java.lang
      .String, int, boolean) (IActivityManager.java:5935)
8 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      content.IContentProvider android.app.ActivityThread.acquireProvider(android.
      content.Context, java.lang.String, int, boolean) (ActivityThread.java:7381)
9 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      content.IContentProvider android.app.ContextImpl$ApplicationContentResolver.
      acquireUnstableProvider(android.content.Context, java.lang.String) (
      ContextImpl.java:3668)
10 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      content.IContentProvider android.content.ContentResolver.
      acquireUnstableProvider(android.net.Uri) (ContentResolver.java:2542)
11 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
      lang.String[], android.os.Bundle, android.os.CancellationSignal) (
      ContentResolver.java:1213)
12 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
      lang.String[], java.lang.String, java.lang.String[], java.lang.String, android
      .os.CancellationSignal) (ContentResolver.java:1161)
13 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at android.
      database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
      lang.String[], java.lang.String, java.lang.String[], java.lang.String) (
      ContentResolver.java:1117)
14 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at java.lang.
      Class java.lang.VMClassLoader.findLoadedClass(java.lang.ClassLoader, java.lang
      .String) (VMClassLoader.java:-2)
15 05-15 21:44:13.493  2247  2247 F ocessService1:0: thread.cc:2517]      at java.lang.
      Class java.lang.ClassLoader.findLoadedClass(java.lang.String) (ClassLoader.
      java:738)
```

```
16 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at java.lang.  
    Class java.lang.ClassLoader.loadClass(java.lang.String, boolean) (ClassLoader.  
    java:363)  
17 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at java.lang.  
    Class java.lang.ClassLoader.loadClass(java.lang.String) (ClassLoader.java:312)  
18 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at android.app.  
    .Service android.app.AppComponentFactory.instantiateService(java.lang.  
    ClassLoader, java.lang.String, android.content.Intent) (AppComponentFactory.  
    java:129)  
19 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.app.ActivityThread.handleCreateService(android.app.  
    ActivityThread$CreateServiceData) (ActivityThread.java:4667)  
20 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.app.ActivityThread.-$Nest$mhandleCreateService(android.app.  
    ActivityThread, android.app.ActivityThread$CreateServiceData) (ActivityThread.  
    java:-1)  
21 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.app.ActivityThread$H.handleMessage(android.os.Message) (ActivityThread.  
    java:2289)  
22 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.os.Handler.dispatchMessage(android.os.Message) (Handler.java:106)  
23 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at boolean  
    android.os.Looper.loopOnce(android.os.Looper, long, int) (Looper.java:205)  
24 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.os.Looper.loop() (Looper.java:294)  
25 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void  
    android.app.ActivityThread.main(java.lang.String[]) (ActivityThread.java:8248)  
26 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at java.lang.  
    Object java.lang.reflect.Method.invoke(java.lang.Object, java.lang.Object[]) (Method.  
    java:-2)  
27 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void com.  
    android.internal.os.RuntimeInit$MethodAndArgsCaller.run() (RuntimeInit.java  
    :552)  
28 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void com.  
    android.internal.os.ChildZygoteInit.runZygoteServer(com.android.internal.os.  
    ZygoteServer, java.lang.String[]) (ChildZygoteInit.java:136)  
29 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void com.  
    android.internal.os.WebViewZygoteInit.main(java.lang.String[]) (WebViewZygoteInit.  
    java:147)  
30 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at java.lang.  
    Object java.lang.reflect.Method.invoke(java.lang.Object, java.lang.Object[]) (Method.  
    java:-2)  
31 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void com.  
    android.internal.os.RuntimeInit$MethodAndArgsCaller.run() (RuntimeInit.java  
    :552)  
32 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at void com.  
    android.internal.os.ZygoteInit.main(java.lang.String[]) (ZygoteInit.java:971)  
33 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] Caused by:  
    android.os.RemoteException: Remote stack trace:  
34 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at com.android.  
    .server.am.ActivityManagerService.enforceNotIsolatedCaller(  
    ActivityManagerService.java:3079)  
35 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at com.android.  
    .server.am.ContentProviderHelper.getContentProvider(ContentProviderHelper.java  
    :130)  
36 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at com.android.  
    .server.am.ActivityManagerService.getContentProvider(ActivityManagerService.  
    java:6782)  
37 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] at android.app
```

```

    .IActivityManager$Stub.onTransact(IActivityManager.java:2776)
38 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517]  at com.android
     .server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2763)
39 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517]
40 05-15 21:44:13.493 2247 2247 F ocessService1:0: thread.cc:2517] (Throwable with
     no stack trace)

```

In AOSP 13, the runtime responds to the changes in the schema by the Management App instantly, whereas in AOSP 14, the response is not consistent. Now trying to go into `MY_match_target_method` of `art/runtime/runtime.cc` and see the flow up close.

The debloating works perfectly when:

```
1 adb uninstall com.atomic.apps.medical.drug.dictionary && adb install success_apps
   /60_com.atomic.apps.medical.drug.dictionary.apk
```

But after closing the app and relaunch, the debloating doesn't work for some methods anymore. Need to see what happens to the runtime when the app is closed and relaunched.

This sandbox related exception has nothing to do with this error as it still shows when the debloating was working perfectly after reinstalling:

```

1 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] No pending
     exception expected: java.lang.SecurityException: Isolated process not allowed
     to call getContentProvider
2 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at java.lang.
     Exception android.os.Parcel.createExceptionOrNull(int, java.lang.String) (
     Parcel.java:3057)
3 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at java.lang.
     Exception android.os.Parcel.createException(int, java.lang.String) (Parcel.
     java:3041)
4 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at void
     android.os.Parcel.readException(int, java.lang.String) (Parcel.java:3024)
5 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at void
     android.os.Parcel.readException() (Parcel.java:2966)
6 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.app
     .ContentProviderHolder android.app.IActivityManager$Stub$Proxy.
     getContentProvider(android.app.IApplicationThread, java.lang.String, java.lang
     .String, int, boolean) (IActivityManager.java:5935)
7 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.
     content.IContentProvider android.app.ActivityThread.acquireProvider(android.
     content.Context, java.lang.String, int, boolean) (ActivityThread.java:7381)
8 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.
     content.IContentProvider android.app.ContextImpl$ApplicationContentResolver.
     acquireUnstableProvider(android.content.Context, java.lang.String) (
     ContextImpl.java:3668)
9 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.
     content.IContentProvider android.content.ContentResolver.
     acquireUnstableProvider(android.net.Uri) (ContentResolver.java:2542)
10 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.
     database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
     lang.String[], android.os.Bundle, android.os.CancellationSignal) (
     ContentResolver.java:1213)
11 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]  at android.
     database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
     lang.String[], java.lang.String, java.lang.String[], java.lang.String, android
     .os.CancellationSignal) (ContentResolver.java:1161)

```

```

12 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at android.
     database.Cursor android.content.ContentResolver.query(android.net.Uri, java.
     lang.String[], java.lang.String, java.lang.String[], java.lang.String) (
     ContentResolver.java:1117)
13 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Class java.lang.VMClassLoader.findLoadedClass(java.lang.ClassLoader, java.lang.
     String) (VMClassLoader.java:-2)
14 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Class java.lang.ClassLoader.findLoadedClass(java.lang.String) (ClassLoader.
     java:738)
15 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Class java.lang.ClassLoader.loadClass(java.lang.String, boolean) (ClassLoader.
     java:363)
16 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Class java.lang.ClassLoader.loadClass(java.lang.String) (ClassLoader.java:312)
17 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at android.app.
     .Service android.app.AppComponentFactory.instantiateService(java.lang.
     ClassLoader, java.lang.String, android.content.Intent) (AppComponentFactory.
     java:129)
18 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.app.ActivityThread.handleCreateService(android.app.
     ActivityThread$CreateServiceData) (ActivityThread.java:4667)
19 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.app.ActivityThread.-$$Nest$chandleCreateService(android.app.
     ActivityThread, android.app.ActivityThread$CreateServiceData) (ActivityThread.
     java:-1)
20 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.app.ActivityThread$H.handleMessage(android.os.Message) (ActivityThread.
     java:2289)
21 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.os.Handler.dispatchMessage(android.os.Message) (Handler.java:106)
22 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at boolean
     android.os.Looper.loopOnce(android.os.Looper, long, int) (Looper.java:205)
23 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.os.Looper.loop() (Looper.java:294)
24 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void
     android.app.ActivityThread.main(java.lang.String[]) (ActivityThread.java:8248)
25 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Object java.lang.reflect.Method.invoke(java.lang.Object, java.lang.Object[])
     (Method.java:-2)
26 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void com.
     android.internal.os.RuntimeInit$MethodAndArgsCaller.run() (RuntimeInit.java
     :552)
27 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void com.
     android.internal.os.ChildZygoteInit.runZygoteServer(com.android.internal.os.
     ZygoteServer, java.lang.String[]) (ChildZygoteInit.java:136)
28 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void com.
     android.internal.os.WebViewZygoteInit.main(java.lang.String[]) (
     WebViewZygoteInit.java:147)
29 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at java.lang.
     Object java.lang.reflect.Method.invoke(java.lang.Object, java.lang.Object[])
     (Method.java:-2)
30 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void com.
     android.internal.os.RuntimeInit$MethodAndArgsCaller.run() (RuntimeInit.java
     :552)
31 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] at void com.
     android.internal.os.ZygoteInit.main(java.lang.String[]) (ZygoteInit.java:971)
32 05-16 01:06:34.222 2319 2319 F ocessService1:0: [thread.cc:2517] Caused by:
     android.os.RemoteException: Remote stack trace:

```

```
33 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] at com.android
     .server.am.ActivityManagerService.enforceNotIsolatedCaller(
     ActivityManagerService.java:3079)
34 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] at com.android
     .server.am.ContentProviderHelper.getContentProvider(ContentProviderHelper.java
     :130)
35 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] at com.android
     .server.am.ActivityManagerService.getContentProvider(ActivityManagerService.
     java:6782)
36 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] at android.app
     .IActivityManager$Stub.onTransact(IActivityManager.java:2776)
37 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] at com.android
     .server.am.ActivityManagerService.onTransact(ActivityManagerService.java:2763)
38 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517]
39 05-16 01:06:34.222 2319 2319 F ocessService1:0: thread.cc:2517] (Throwable with
     no stack trace)
```

7.5 May 19, 2024

7.5.1 What makes up FjordPhantom exactly?

Hooking

Hook into APIs that is used to determine if Accessibility services are turned on, and return false information to bypass screenreader and rooting detection. Can also hook to UI functionalities such as dialog boxes to close them immediately if they contain certain strings.

Usage of hooking in Malware analysis

Paper	Hook Logic
Without System Modification	
Boxify, USENIX2015 ³	Secure sandboxing by dynamically loading and executing within its own process.
Malton, USENIX2017 ⁴	No firmware or application modification, or rooting needed. Non-invasive Multi-layer (Java, C++ and ARM) Taint Analysis and Path Exploration.
ARTDroid, IMPS2016 ⁶	Taint propagation at Instruction (ARM) layer using Valgrind ⁵ because both app and framework code are compiled to native code in the ART runtime.
RetroSkeleton, biSys'13 ⁷	No modification needed for app and Android system. Find target methods in virtual memory, hijack vtable, and set native hooks.
Mo-	App re-writing framework by hooking apps using static instrumentation, for unmodifiable Android devices. Uses <i>transformation policies</i> to replace <i>target</i> methods with <i>handler</i> methods.

³<https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-backes.pdf>

⁴<https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-xue.pdf>

⁵<https://valgrind.org/docs/valgrind2007.pdf>

⁶https://ceur-ws.org/Vol-1575/paper_10.pdf

⁷<https://www.cs.ucdavis.edu/~hchen/paper/mobisys2013rs.pdf>

Paper	Hook Logic
AppGuard, TACAS2013 ⁸	Hooking apps by static instrumentation to invoke inline reference monitoring and enforcing security and privacy policies. No need for firmware modification, or root access.
With System Modification	
Ronin, DASC2018 ⁹	Hook Main activity to call Ronin Ronin overwrites PLT tables ¹⁰ for hooking native functions in the memory
StaDynA, CODASPY'15 ¹¹	Hooking framework APIs to log classes and methods (path to code file and stack trace) loaded dynamically and via reflection that cannot be inferred by static analysis.
TaintDroid, USENIX2010 ¹²	Hooking taint sources such as sensor manager, information databases and device identifiers. Also hooking taint sinks (native socket library for network connections)
DroidScope, USENIX2012 ¹³	Runtime and kernel instrumentation for dynamic taint analysis on Java- and OS-level
Andrubis, Technical Report ¹⁴	Runtime instrumentation

Table 7.3: Permission Denied Syntax⁸https://link.springer.com/content/pdf/10.1007/978-3-642-36742-7_39.pdf⁹https://www.researchgate.net/publication/328611051_Android_Hooking_Revisited¹⁰<https://www.technovelty.org/linux/plt-and-got-the-key-to-code-sharing-and-dynamic-libraries.html>¹¹https://lilicoding.github.io/SA3Repo/papers/2015_zhauniarovich2015stadyna.pdf¹²https://static.usenix.org/event/osdi10/tech/full_papers/Enck.pdf¹³<https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final107.pdf>¹⁴<https://www.vvdveen.com/publications/TR0414001.pdf>

7.6 May 22-27, 2024

7.6.1 Repackaging

Apps with malicious payloads added to apps on non-official app markets. Most popular code injection method is to repackage with additional code. But can be easily detected.

Name	Description
Repackaging Attack on Android Banking Applications and Its Countermeasures, WPC'13 ¹⁵	Example of repackaging attack by getting rid of antivirus instance-checking and modifying integrity-checking logic to send hash of original APK instead of the modified APK Countermeasures: Self-signing restriction, Code obfuscation, and Code attestation
Detecting Repackaged Android Apps: Lit. Review and Benchmark, TSE2021 ¹⁶	Detection approaches: Similarity computation (Code fuzzy hash, layout tree, resource files, layout group graph, method statements, control flow graph, identifiers, APIs, Intents, Permissions, Sensors, method signature, inputs/outputs of methods, control flow pattern, user interfaces, ICC-based view graph, data-dependency graph, abstract syntactic tree, API call sequences, etc...), Runtime monitoring (execution trace, virtualization-based protection, watermarking, and package name), Supervised learning (user interactions, sensitive APIs, permissions, system call sequences, distance of sensitive subgraphs), Unsupervised learning (program dependency graph, permissions, intents, components, API calls, ICC, etc...), and Sympton discovery (string offset order)
Active Warden Attack: On the (In)Effectiveness of Android App Repackage-Proofing, DSC2022 ¹⁷	By passing repackaging-proofing using Active Warden Attacks which is basically MITM (via customized cached local binder proxy) that intercepts and relay integrity verification messages between the victim app and the Android Framework APIs Existing repackaging-proofing works have flawed assumption of trusting ICCs

¹⁵<https://link.springer.com/article/10.1007/s11277-013-1258-x>

¹⁶<https://ieeexplore.ieee.org/document/8653409>

¹⁷https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=7706&context=sis_research

Name	Description
Rewrap Me If You Can, ACSAC'21 ¹⁸	<p>Existing anti-repacking and anti-virtualization solutions don't work. Virtualization eliminates the need to modify and repack the original APK. The container can execute any external app without modifying it, intercept all the API invocations to the Android OS and tamper the responses.</p> <p>Protects the apps using code splitting (removing portion of code from the original app to split the original APK into multiple DEXes), and IAT (injection of controls which are evaluated during the interaction between the container and the plugin).</p>

Table 7.4: Papers concerning Repackaging Attacks

7.6.2 Virtualization and Containers

Allows installation of the same app multiple times to be used with different accounts¹⁹. Those different apps run in the same sandbox to allow them to access each other's files and memory to debug each other. Allows breaking of sandbox without rooting. Loads additional code (logic of malware and invocations of hooking framework) into the process of hosted app.

Name	Description
VAHunt: Warding Off New Repackaged Android Malware in App-Virtualization's Clothing, CCS'20 ²⁰	<p>App-level virtualization where the host app creates virtual environment by dynamic proxy, and uses API hooking (so that the system thinks that all requests come from the host app) and binder proxy to bypass system service restrictions</p> <p>Stateful detection model using FSM and hooking intent APIs to detect app-level virtualization</p> <p>Data flow analysis to extract self-hiding loading strategies to differentiate between malicious and benign app-level virtualization-based apps</p>

¹⁸<https://dl.acm.org/doi/pdf/10.1145/3485832.3488021>

¹⁹<https://promon.co/security-news/fjordphantom-android-malware/>

²⁰<https://par.nsf.gov/servlets/purl/10220267>

Name	Description
Seamless In-App Ad Blocking on Stock Android, SPW'2017 ²¹	App-level virtualization for deploying security solutions (stripping advertising related-code without breaking app) No modification to both system or app (apart from removing)
HideMyApp: Hiding the Presence of Sensitive Apps on Android, USENIX'2019 ²²	Built on top of app-level virtualization, the container app launches the APK file of the sensitive app within its context so that the sensitive app is not registered with the OS to protect the app from nosy apps. No modification to firmware; few modification to the app
Anti-Plugin, hat'2017 ²³	Inspection of plugin technology (app-level virtualization) abuse by malware Malware can update itself without rooting the phone or having to repackage; they only need to be downloaded and installed by the user the first time Malware can evade static analysis as the code is updated only after the installation
App in the Middle, MACS'2019 ²⁴	App-level virtualization threat analysis Can detect virtualization by detecting instrumentation and verifying file system structure
Parallel Space Travelling, SACMAT'20 ²⁵	App-level virtualization consists of Virtual Framework (for emulation of core system services like PackageManager and ActivityManager) and Hook module (for intercepting the interaction between the guest apps and the system services). Either GuestApp (privilege escalation, code injection, ransomware, phishing, cloning) or HostApp (hijacking, antivirus evasion) can be malicious. Can detect by inspecting private directory info, call stack, and memory maps

²¹<https://svenbugiel.de/publication/wissfeld-17-most/wissfeld-17-most.pdf>

²²<https://www.usenix.org/system/files/sec19-pham.pdf>

²³https://paper.bobylive.com/Meeting_Papers/BlackHat/Asia-2017/

²⁴<https://dl.acm.org/doi/pdf/10.1145/3322205.3311088>

²⁵<https://www.cs.ucr.edu/~heng/pubs/sacmat2020.pdf>

Name	Description
VPBox, CCS'21 ²⁶	<p>Modify Android Framework for sandboxing via container-based virtualization which uses Linux container tools transplanted to Android or runs another Android OS as guest OS by mounting virtual file system and runtime.</p> <p>Different from app-level virtualization where host app provides virtualization on top of the Android framework and create system service proxies to intercept messages. The host app has to hook the API invocations of the guest app so that the Android system thinks that all API requests are from the host apps, leaving host app's signatures in the guest app's call stack and memory.</p>
Condroid, Cloud2015 ²⁷	<p>Mobile-OS-level virtualization using containers by modifying Android Framework (virtual Binder IPC mechanisms)</p> <p>Runs containers which are virtual machines running an isolated Android system with pre-installed apps. Android framework inside the Android systems inside the containers are modified.</p> <p>Linux kernel reconfigured to enable cgroups, namespace features, creating virtual devices and virtual device drivers.</p> <p>Android Framework of host Android system modified to cooperate with the containers.</p> <p>LXC tools transplanted from Linux to Android runtime environment for creating and managing containers.</p>

Table 7.5: Papers concerning App-virtualization

²⁶<https://par.nsf.gov/servlets/purl/10327186>

²⁷http://arc.zju.edu.cn/_upload/article/files/d9/bd/a5f57e3a4143b907382976333a87/d086a9aa-13c0-4261-9109-b63686c62c97.pdf

Chapter 8

June

8.1 June 10, 2024

8.1.1 Decompile APKs

JADX

First Decompiler to try is JADX¹.

Need to install JDK11:

```
1 sudo apt install openjdk-11-jdk
2 update-java-alternatives --list
3 sudo update-alternatives --config java
```

Do not install JRE sudo apt install openjdk-11-jre-headless, as it is not JDK and lack some development library.

APK Tool²

Dex2Jar³

8.1.2 Repackaging Attack

Extract APK from Installed App

Download and install Screen Logger APK⁴:

```
1 adb install Screen\ Logger_1.1_APKPure.apk
```

After installation, grant requested permissions to the Screen Logger app for it to run properly.

Install DBS mobile banking application from Google Playstore, and extract its APK:

¹<https://github.com/skylot/jadx>

²<https://github.com/iBotPeaches/Apktool>

³<https://github.com/pxb1988/dex2jar>

⁴<https://apkpure.com/screen-logger/com.ultra.app.screenlogger/download>

```

1 adb shell pm list packages -f -3
2 adb pull /data/app/~~VDIXkS0BT9AdrdsivBoitg==/com.dbs.sg.dbsmbanking-
  b3QDPSklmc7YrFj8n02hHA==/base.apk

```

Safeguards

Emulator Detection logic⁵ in OCBC app:

```

1 public class Validation {
2     public static boolean isEmulator() {
3         String str = Build.FINGERPRINT;
4         if (!str.startsWith("generic") && !str.startsWith("unknown")) {
5             String str2 = Build.MODEL;
6             if (!str2.contains("google_sdk") && !str2.contains("Emulator") && !
7                 str2.contains("Android.SDK.built.for.x86") && !Build.MANUFACTURER.
8                 contains("Genymotion") && (!(Build.BRAND.startsWith("generic") ||
9                 Build.DEVICE.startsWith("generic")) && !"google_sdk".equals(Build
10                .PRODUCT))) {
11                     return false;
12                 }
13             }
14         }
15     return true;
16 }

```

Root check when loading OneTouchActivity:

```

1 public boolean isRooted() {
2     if (!Validation.isRooted(this)) {
3         return false;
4     }
5     new AlertDialog.Builder(this).setTitle("").setMessage(getString(R.string.
6         message_no_root_support)).setCancelable(false).setPositiveButton("OK", new
7         DialogInterface.OnClickListener() { // from class: com.ocbc.mdt.onetouch.
8             activity.OneTouchActivity.3
9             @Override // android.content.DialogInterface.OnClickListener
10            public void onClick(DialogInterface dialogInterface, int i11) {
11                OneTouchActivity.this.finish();
12                Process.killProcess(Process.myPid());
13            }
14        }).show();
15    return true;
16 }

```

String constant that stores the warning message:

```

1 public static final String DEFAULT_ROOT_MSG_SG = "We cannot proceed with OCBC
  mobile banking as we suspect your device is 'jailbroken', 'rooted' or running
  a non-compatible operating system. This leaves you vulnerable to fraud
  attacks. Please login to www.ocbc.com to continue banking.";

```

Root detection checks for multiple conditions to be true:

```

1 public boolean n() {
2     return j() || h() || b("su") || c() || e() || l() || g() || f() || d();
3 }

```

One of the Root detection logic uses RootBeer⁶:

⁵<https://ray-chong.medium.com/android-emulator-detection-4d0f994aab5e>

⁶<https://github.com/scottyab/rootbeer>

```
1 public boolean f() {
2     if (!a()) {
3         n50.a.a("We could not load the native library to test for root");
4         return false;
5     }
6     int length = a.f55724d.length;
7     String[] strArr = new String[length];
8     for (int i11 = 0; i11 < length; i11++) {
9         strArr[i11] = a.f55724d[i11] + "su";
10    }
11    RootBeerNative rootBeerNative = new RootBeerNative();
12    try {
13        rootBeerNative.setLogDebugMessages(this.f55727b);
14        return rootBeerNative.checkForRoot(strArr) > 0;
15    } catch (UnsatisfiedLinkError unused) {
16        return false;
17    }
18 }
```

Accessibility Detection

8.1.3 App Virtualization Attack

