ACM Asia Conf. on Comp. and Comm. Security (**AsiaCCS**), Auckland, Jul 2019

Towards Understanding Android System Vulnerabilities: Techniques and Insights

Daoyuan Wu[1], Debin Gao[1], Eric K. T. Cheng[2],
Yichen Cao[3], Jintao Jiang[3], and Robert H. Deng[1]

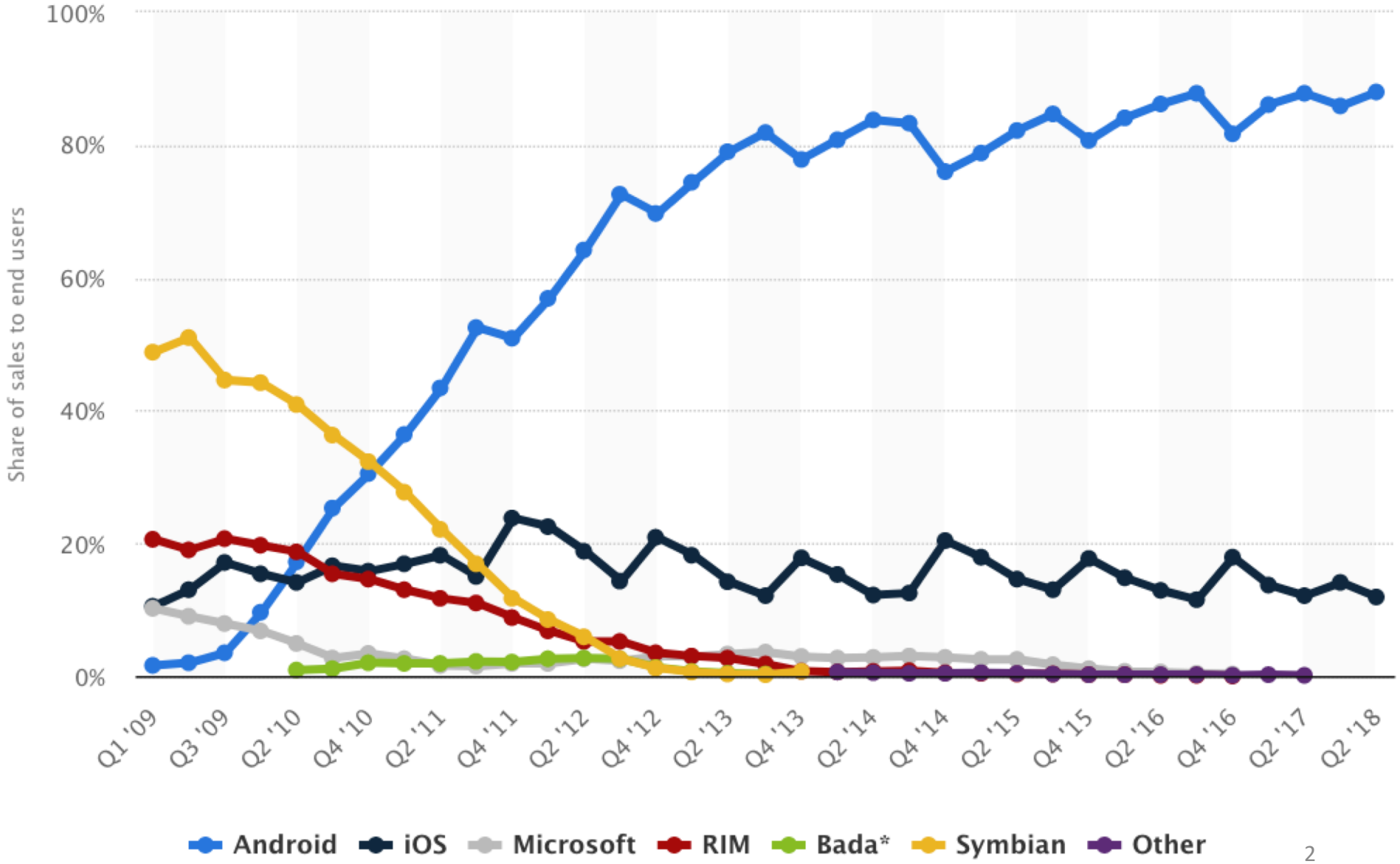
1 SMU SINGAPORE MANAGEMENT UNIVERSITY

2 THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

3 SOBUG 安全众测平台

Android has become the most popular system

A global market share at over **80%** since 2013



https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

More and more attacks targeted at Android

New AndroRAT Exploits Dated Privilege Escalation Vulnerability, Allows Permanent Rooting

Posted on: February 13, 2018 at 12:00 am **Posted in:** Malware, Mobile, Vulnerabilities
Author: Mobile Threat Response Team

10

by Veo Zhang, Jason Gu, and Seven Shen

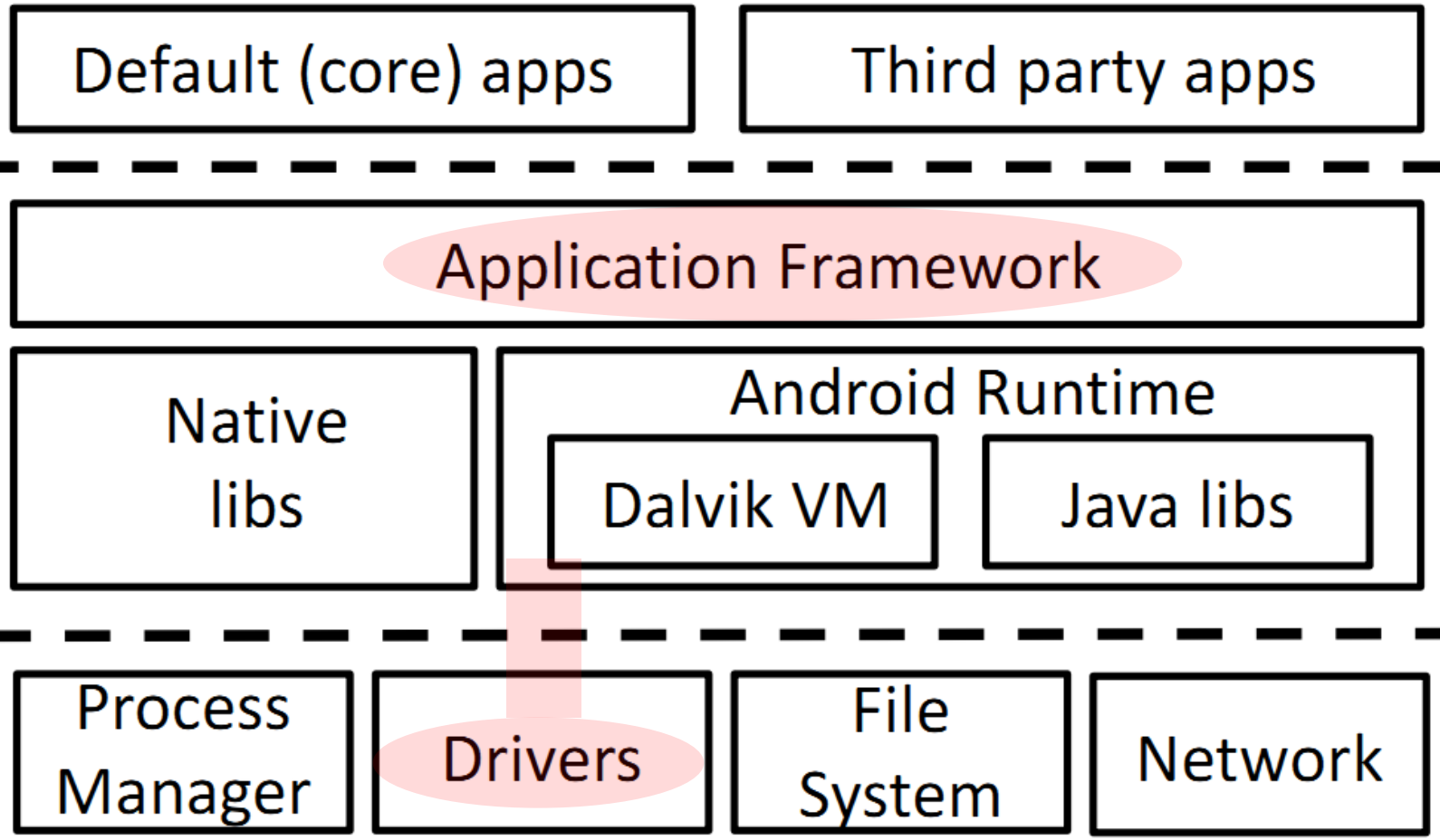
Trend Micro detected a new variant of Android Remote Access Tool (AndroRAT) (identified as ANDROIDOS_ANDRORAT.HRXC) that has the ability to inject root exploits to perform malicious tasks such as silent installation, shell command execution, WiFi password collection, and screen capture. This AndroRAT targets CVE-2015-1805, a publicly disclosed vulnerability in 2016 that allows attackers to penetrate a number of older Android devices to perform its privilege escalation.

RATs have long been a common Windows threat, so it shouldn't be a surprise that it has come to Android. A RAT has to gain root access — usually by exploiting a vulnerability — in order to have control over a system. Discovered in 2012, the original authors intended AndroRAT — initially a university project — as an open-source client/server application that can provide remote control of an Android system, which naturally attracted cybercriminals.

Prior Arts in analyzing Android vulnerabilities

Woodpecker [NDSS'12] CHEX [NDSS'12] SSLMalloDroid [CCS'12] CryptoLint [CCS'13]

FileAtk [ISC'14] CredMiner [WiSec'15] UnixSocket [CCS'16] XAWI [CCS'17] OSV [S&P'18]



App level extensively studied

System level much less explored in the literature

- Mostly on framework issues
  - [CCS'15], [NDSS'16], [NDSS'18] …
- Specific drivers:
  ION [CCS'16] Binder [ACSAC'16]
- And their exposed interfaces:
  [S&P'14] and [USENIX SEC'18]

Google maintained a new source for white-hats to report Android system vulnerabilities

Android Security Bulletin program

2,179 vulnerabilities reported over around three years (08/2015 -- 06/2018)

Bulletins

Bulletin	Languages	Published date	Security patch level
June 2019	English / 日本語 / 한국어 / рýсский / 简体中文 / 繁體中文 (台灣)	June 3, 2019	2019-06-01 2019-06-05
May 2019	English / 日本語 / 한국어 / рýсский / 简体中文 / 繁體中文 (台灣)	May 6, 2019	2019-05-01 2019-05-05
April 2019	English / 日本語 / 한국어 / рýсский / 简体中文 / 繁體中文 (台灣)	April 1, 2019	2019-04-01 2019-04-05

Could we effectively mine these vulnerabilities for insights?

Outline

- Background and Objectives
- Our analysis framework
- Some interesting results

A sample webpage of Android Security Bulletin

Elevation of privilege vulnerability in ServiceManager

An elevation of privilege in ServiceManager
arbitrary services that would normally be pro
This issue is rated as High severity due to th

CVE	References	Severity
CVE-2016-3900	A-29431260 [2]	High

Elevation of privilege vulnerability in

An elevation of privilege vulnerability in Lock
to clear the device PIN or password. This iss
interaction requirements for any developer o

CVE	References	Severity
CVE-2016-3908	A-30003944	High

Contents

```
ServiceManager: Restore basic uid check

Prevent apps from registering services without relying on selinux checks.

Bug: 29431260

Change-Id: I38c6e8bc7f7cba1cbd3568e8fed1ae7ac2054a9b
(cherry picked from commit 2b74d2c1d2a2c1bb6e9c420f7e9b339ba2a95179)
```

Commit description

```
diff --git a/cmds/servicemanager/service_manager.c b/cmds/servicemanager/service_manager.c
index 21fdff0..8a8e688 100644
--- a/cmds/servicemanager/service_manager.c
+++ b/cmds/servicemanager/service_manager.c
```

Patched code file(s)

```
@@ -121,6 +121,11 @@
 static int svc_can_register(const uint16_t *name, size_t name_len, pid_t spid, uid_t uid)
 {
     const char *perm = "add";
+
+    if (uid >= AID_APP) {
+        return 0; /* Don't allow apps to register services */
+    }
+
     return check_mac_perms_from_lookup(spid, uid, perm, str8(name, name_len)) ? 1 : 0;
 }
```

Detailed code fragments

Analysis Objectives

platform/system/bt/bta/dm/bta.cc

Modules of Vulnerabilities

Patch Code Complexity

Patch Code Patterns

<h3 id="eopv-in-servicemanager">

Shed light on the system modules that are susceptible and require more security attention.

Implementation bugs can be an important source of Android system vulnerabilities.

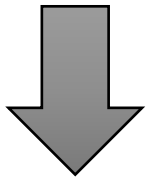
These patterns can be leveraged for automatic vulnerability detection.

Need a database structure that can store all the text and code information in an organized and searchable structure.

Designing a Hierarchical Database Structure

	Month	Location	CveID	RefID	VulType	Level	Version	Date	RefUrl
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
86	2016-10-01	servicemanager	CVE-2016-3900	A-29431260 [2]	EoP	High	5.0.2, 5.1....	Jun 15, 2016	https://andro...

One vulnerability record in the metadata DB



“add” ”del”
“ctx” “hunk”

VID	RefUrl	Module	DiffCode	SubText	MsgText	LineNum
..	Filter	Filter	Filter	Filter	Filter	Filter
86	https://android.goo...	platform/frameworks/native	{"cmds/servicemanager/servi...	ServiceM...	Prevent ap...	3
86	https://android.goo...	platform/frameworks/native	{"cmds/servicemanager/Andr...	ServiceM...	This should...	2

Two corresponding records in the patch code DB


```
{"cmds/servicemanager/Android.mk":[{"line":1,
"code":[["D","LOCAL_SHARED_LIBRARIES := liblog libselinux"],
["A","LOCAL_SHARED_LIBRARIES := liblog libcutils libselinux"]]}],
"cmds/servicemanager/service_manager.c":[{"line":1, "code":[["D","if (uid >= AID_APP) {"], ["A","if (multiuser_get_app_id(uid) >= AID_APP) {"]]}]}
```


One or more code fragments in each JSON block

searchable

```
select median ( json_array_length(value) )
from PatchTable, json_each(PatchTable.DiffCode)
where PatchTable.DiffCode like '{%}' and key not like '%.s';
```

A robust method to study the complexity of patch code

Must exclude auxiliary code lines: blank, import/include, and comment lines.

```
-    /* i2_pic_width_in_luma_samples and i2_pic_height_in_luma_samples
-       should be multiples of min_cb_size. Here these are aligned to 8,
-       i.e. smallest CB size */
-    ps_sps->i2_pic_width_in_luma_samples = ALIGN8(ps_sps->i2_pic_width_in_luma_samples);
-    ps_sps->i2_pic_height_in_luma_samples = ALIGN8(ps_sps->i2_pic_height_in_luma_samples);
-
```

countFrag = max(countAdd, countDel)

```
- (ps_sps->i1_log2_ctb_size > 6))
+ (ps_sps->i1_log2_ctb_size > 6) ||
+ (ps_sps->i2_pic_width_in_luma_samples % (1 << ps_sps->i1_log2_min_coding_block_size) != 0))
```

countFrag = 2

countFile = sum(countFrag)

Automatically Clustering Patch Code Patterns

Diff Code Fragments

{"cmds/servicemanager/Android.mk":[{"line":1, "code":[["D","LOCAL_SHARED_LIBRARIES := liblog libselinux"], ["A","LOCAL_SHARED_LIBRARIES := liblog libcutils libselinux"]]}], "cmds/servicemanager/service_manager.c":[{"line":1, "code":[["D","if (uid >= AID_APP) {"], ["A","if (multiuser_get_app_id(uid) >= AID_APP) {"]]}]}

Extract essential changes

uint8_t --> uint32_t
uint8_t --> uint16_t
writeLong --> writeInt
…
--> = 0
if --> if || value <= 0
%p --> %pK

	Change
	writeLong --> writeInt
	uint8_t --> uint32_t
t *pDest, uint8_t *pSrc, uint8_t len); t *pDest, uint8_t *pSrc, uint16_t len);	uint8_t --> uint16_t
= getCameraService(); getCameraService();	sp<ICameraService>& --> sp<ICameraService>
, __func__, ctrl->ndx, ctrl->base); pK', __func__, ctrl->ndx, ctrl->base);	%p --> %pK

Calculate pairwise similarity

```
[[ 1.          0.96774193 ..., 0.67603485]
 [ 0.96296296  1.         ..., 0.68240740]
 [ 0.97530864  0.95238095 ..., 0.68954248]
                          ...,
 [ 0.58308895  0.63878788 ..., 0.99649122]
 [ 0.59872153  0.59206192 ..., 1.        ]
 [ 0.57966764  0.56245791 ..., 0.99649122]]
```

Generate clusters via affinity propagation **[Science'07]**

Cluster 1
uint8_t --> uint32_t
uint8_t --> uint16_t
…
uint8_t --> uint16_t

…

Cluster N
%p --> %pK
%p --> %pK
%p --> %pK
%p --> %pK
…
%p --> %pK
%p --> %pK

Dataset and Vulnerability Metadata

2,179 vulnerabilities; 1,349 publicly available patches

	RCE	EoP	ID	DoS	N/A	
Critical	200	156	0	0	82	438
High	47	641	112	133	402	1,335
Moderate	6	156	197	19	14	392
Low	1	1	4	8	0	14
	254	954	313	160	498	

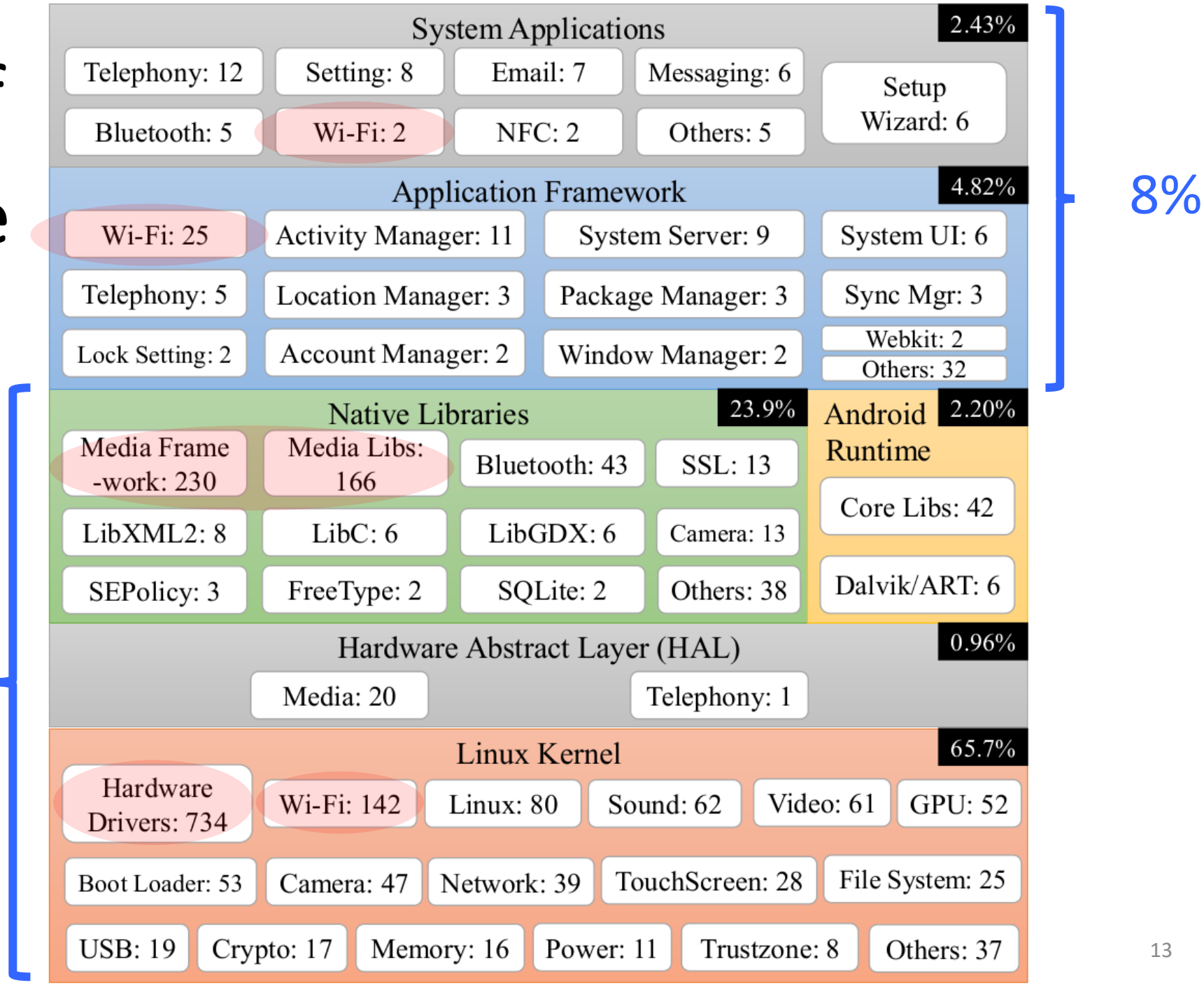
81% (1,773)

55% (1,208)

+23%

Notes:
RCE = Remote Code Execution; EoP = Elevation of Privilege;
ID = Information Disclosure; DoS = Denial of Service.
N/A = Not Available, due to closed-source driver components.

Analysis of Vulnerable Modules

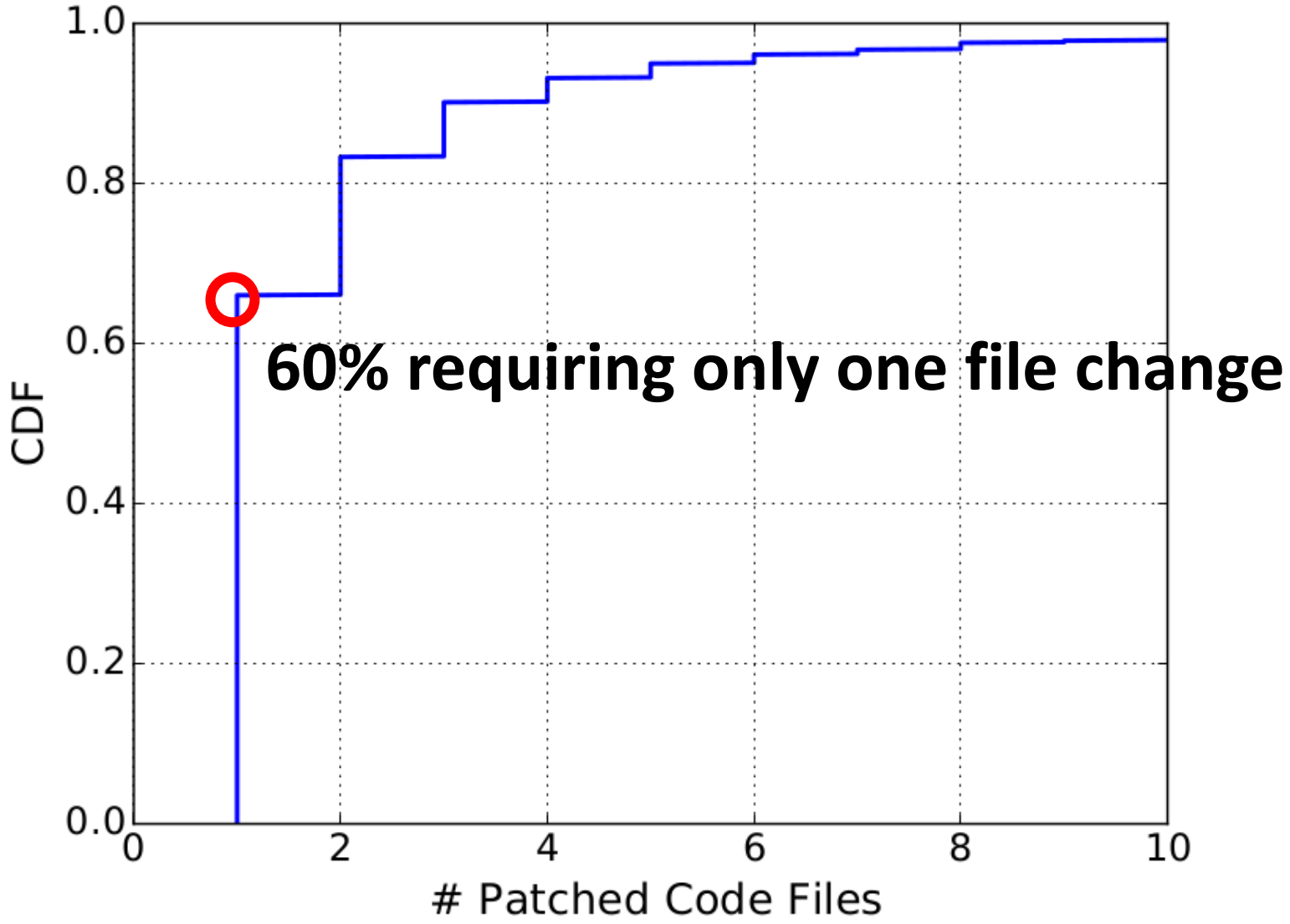


Code that was frequently reported vulnerable

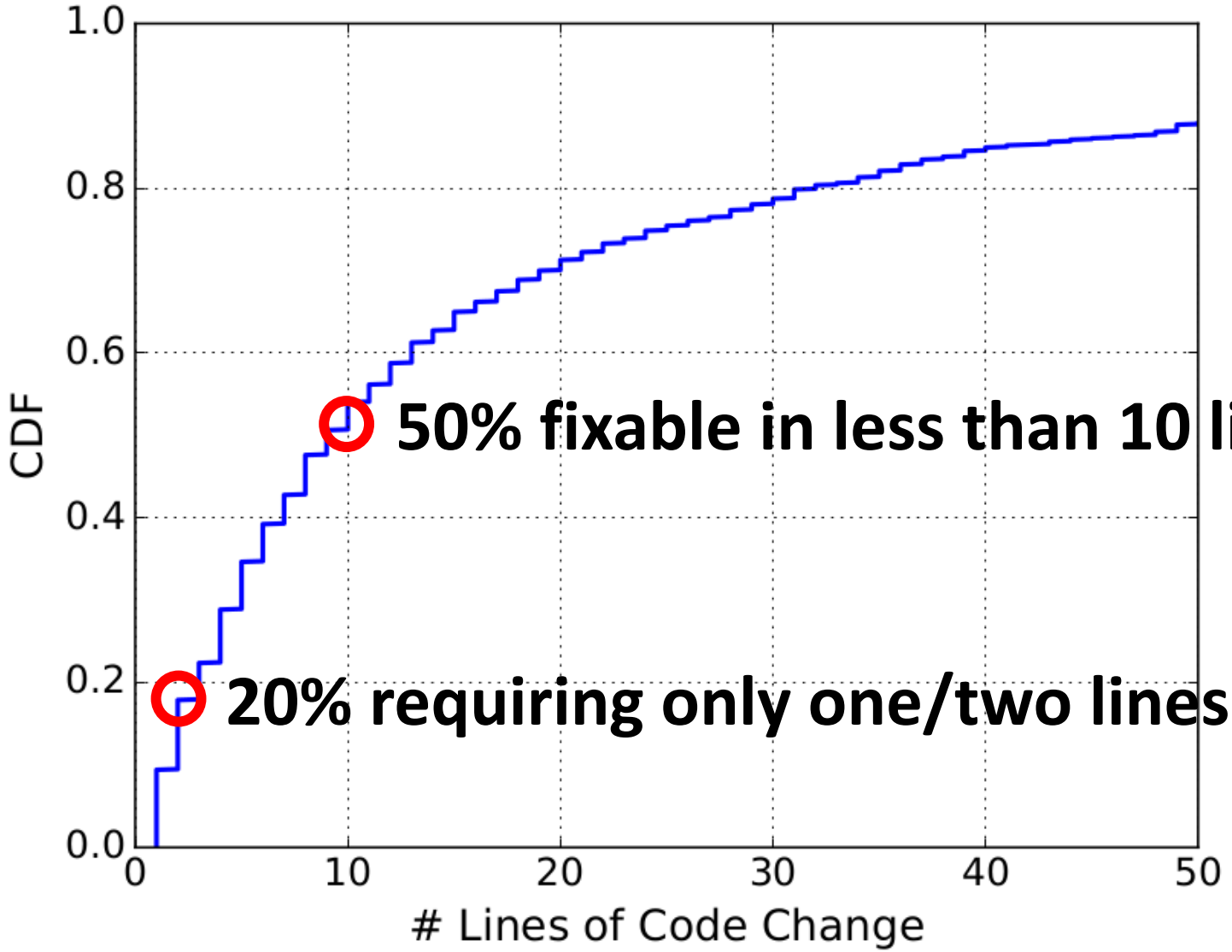
Code	#
media/libstagefright/MPEG4Extractor.cpp	26
decoder/ih264d_parse_pslice.c	23
decoder/ih264d_api.c	20
decoder/ih264d_parse_slice.c	17
drivers/misc/qseecom.c	17
media/libeffects/lvm/wrapper/Bundle/EffectBundle.cpp	17
CORE/HDD/src/wlan_hdd_cfg80211.c	15
app/aboot/aboot.c	14
decoder/ihevcd_parse_headers.c	14
services/audioflinger/Effects.cpp	14
decoder/impeg2d_dec_hdr.c	13
decoder/ih264d_parse_headers.c	11
com/android/server/am/ActivityManagerService.java	11
post_proc/equalizer.c	10

Can help developers avoid making similar mistakes in the same module or code

Analysis of Patch Code Complexity



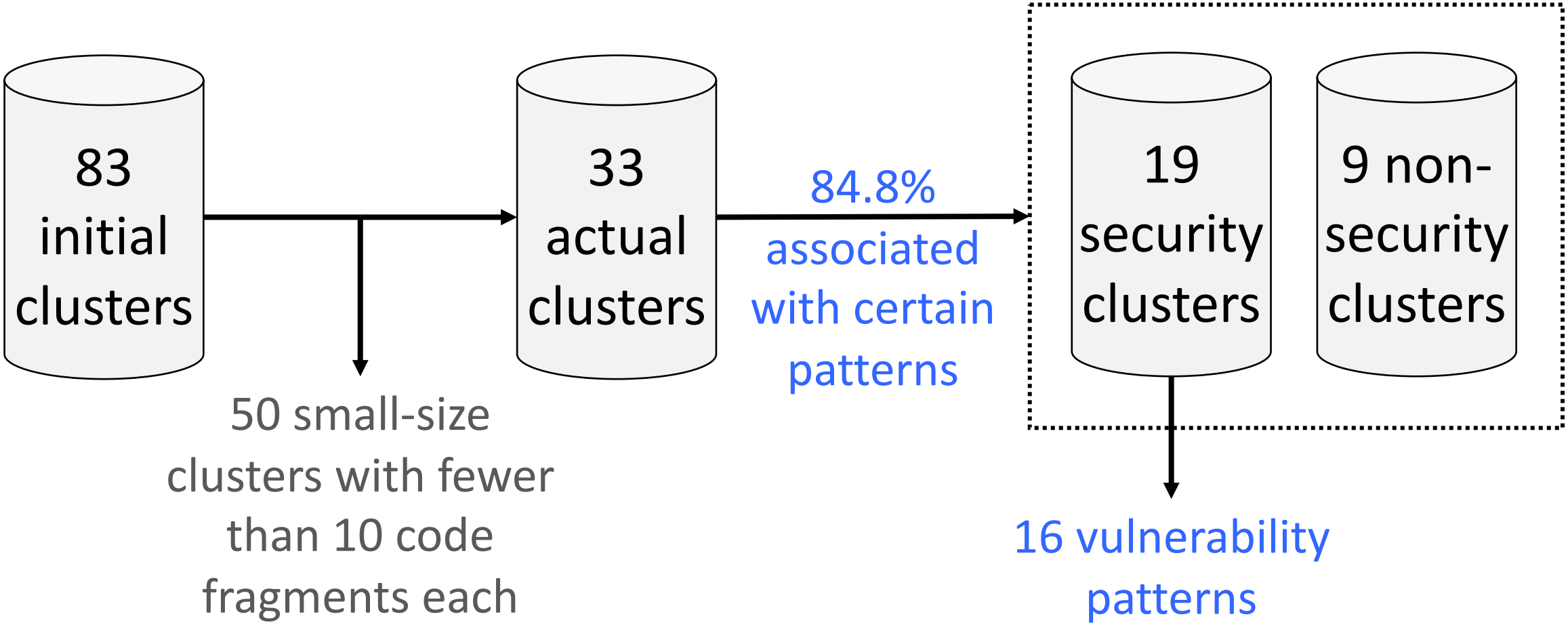
Analysis of Patch Code Complexity (Cont'd)



50% fixable in less than 10 lines of code

20% requiring only one/two lines of code

Intermediate results of our pattern clustering



16 Clustered Patterns for Android System Vulns

ID	Description	Pattern (using diff code's essential change format)	Example	Known?	Cluster ID
P1	Kernel address leakage due to using %p	`%p --> %pK`	C5	✗	73
P2	Mis-retrieving Android service by reference	`sp<XXXService>& --> sp<XXXService>`	C4	✗	21
P3	Inconsistent Android Parcelable serialization	`writeLong --> writeInt OR + writeInt();`	C1	✗	21, 81
P4	Mis-exported component in system apps	`exported='true' --> exported='false'`	–	✔ [66]	21
P5	Missing or mis-setting IF check condition	`if [OLD_CONDITION] --> if NEW_CONDITION`	C9	◑ [19]	63, 2
P6	Use-after-free and double free issues	`+/- XXX_free();`	–	✔ [32]	1
P7	Missing Android permission/UID checking	`+ checkXXXPermission()/checkCallerXXX();`	–	◑ [59]	1
P8	Overflow due to inappropriate #define value	`#define INT1 --> #define INT2`	C10	✔ [19]	74
P9	Incomplete C++ destruction	`+ virtual void onReset();`	–	✗	14
P10	Uninitialized data due to missing memset()	`+ memset();`	–	✔ [52]	50, 36
P11	Uninitialized data due to unassigned variable	`VARIABLE --> VARIABLE = INIT_VALUE`	C6	✔ [52]	32, 4, 15
P12	Missing certain parameter, causing logic flaws	`--> , PARAMETER`	C7	✗	52
P13	Overflow due to missing error case checking	`+ if (CONDITION) + { return ERROR; }`	–	◑ [19]	31
P14	Forgetting to set certain variable const/transient	`--> const / transient`	–	✗	58
P15	Integer overflow due to inappropriate INT type	`uint8_t|int --> uint16|32_t|long|size_t`	C2, C3	✔ [64]	7, 12
P16	Data race due to missing lock/unlock	`+ XXX_lock(); + XXX_unlock();`	–	✔ [25]	69

Six new patterns: P1, P2, P3, P9, P12, P14

Two more Android-specific patterns: P4, P7

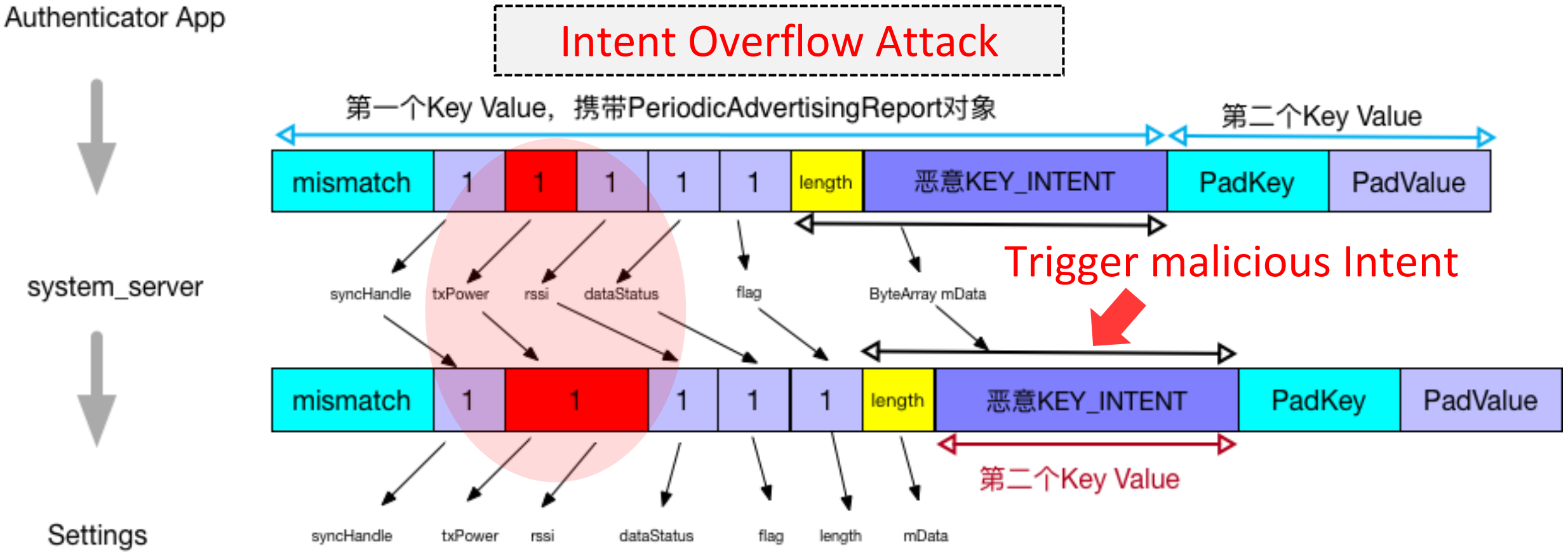
P3: Inconsistent Android Parcelable serialization

CVE-2017-13315:

telephony/java/com/android/internal/telephony/DcParamObject.java

CVE-2017-13288:

core/java/android/bluetooth/le/PeriodicAdvertisingReport.java



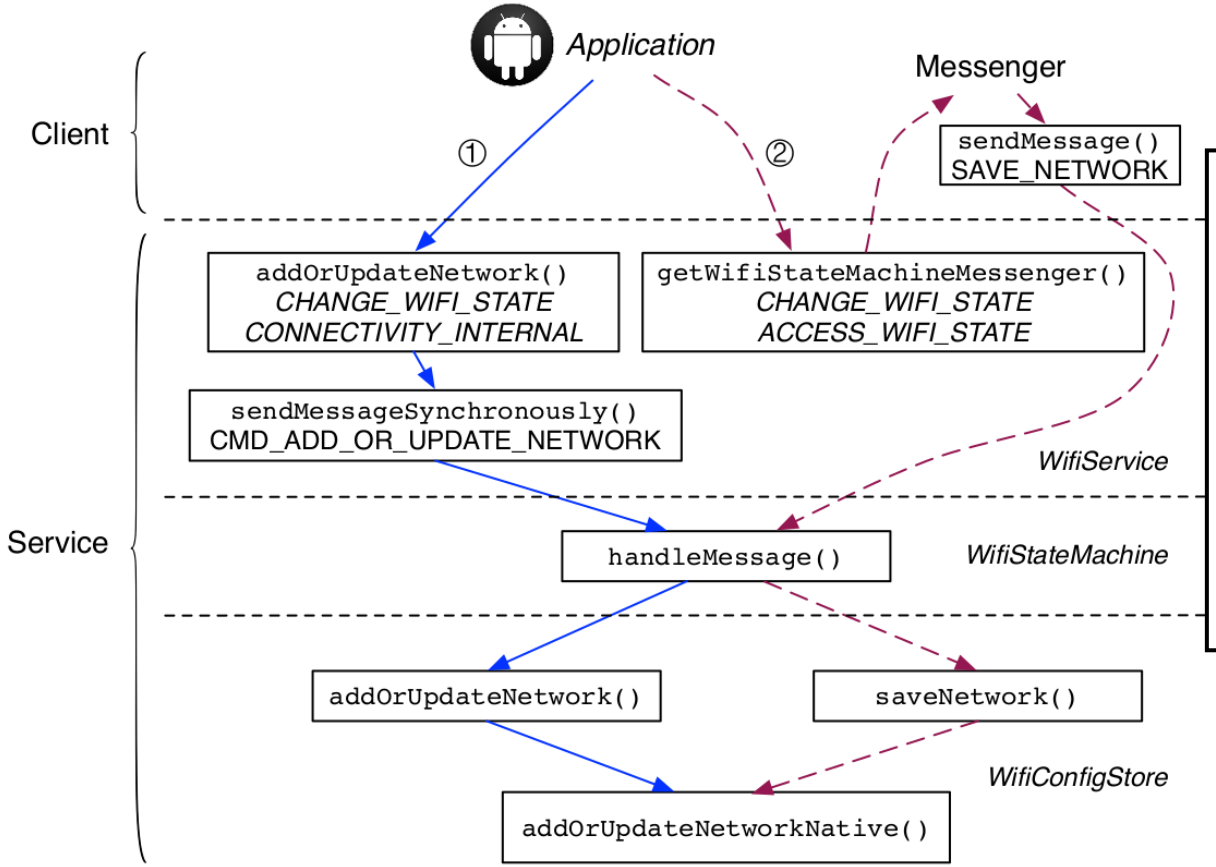
http://www.ms509.com/2018/07/03/bundle-mismatch/

P7: Missing Android permission/UID checking

Kratos [NDSS'16]

AceDroid [NDSS'18]

Only for the framework-level Java code



CVE-2017-13236:
keystore/key_store_service.cpp
(native-level C/C++ code)

```
-   if (!checkBinderPermission(P_GEN_UNIQUE_ID))
+   if (!checkBinderPermission(P_GEN_UNIQUE_ID) ||
      originalUid != IPCThreadState::self()->getCallingUid())
{
        return ResponseCode::PERMISSION_DENIED;
}
```

Conclusion and Future Work

- Conducted the first systematic study of Android system vulnerabilities by analyzing all 2,179 vulnerabilities and their 1,349 publicly available patches on the Android Security Bulletin program over around three years.

- Proposed an analysis framework and its three analyzers, including the novel similarity-based clustering algorithm, to:
  - Pinpoint the modules of Android vulnerabilities;
  - Study the complexity of Android patch code;
  - Obtain 16 vulnerability patterns, including six new ones not in the literature.

- **Future work**: Improve our clustering algorithm to support long code fragments, because the current version is limited to short code fragments only.

Contact: Daoyuan Wu
Twitter @**dao0x**