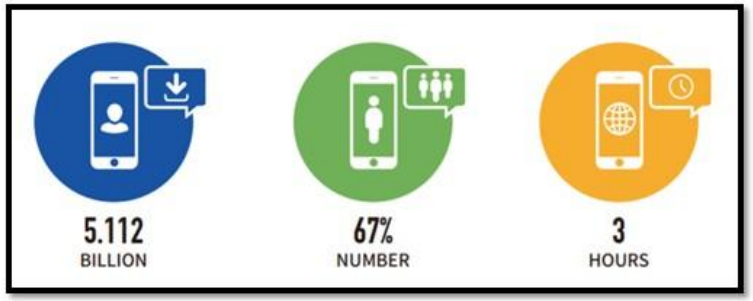
# ICC Leak and Taint Analysis for Android App

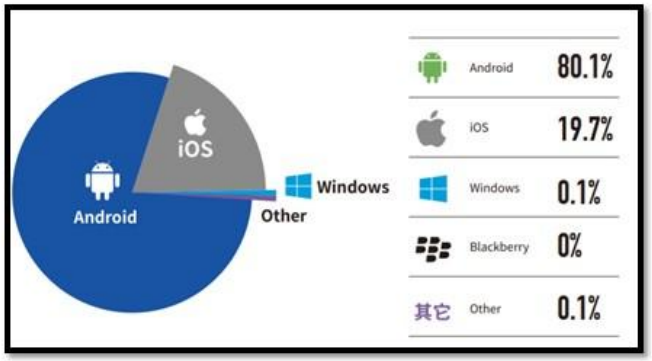
Chenkai Guo
guochenkai@nankai.edu.cn
2022.11.20

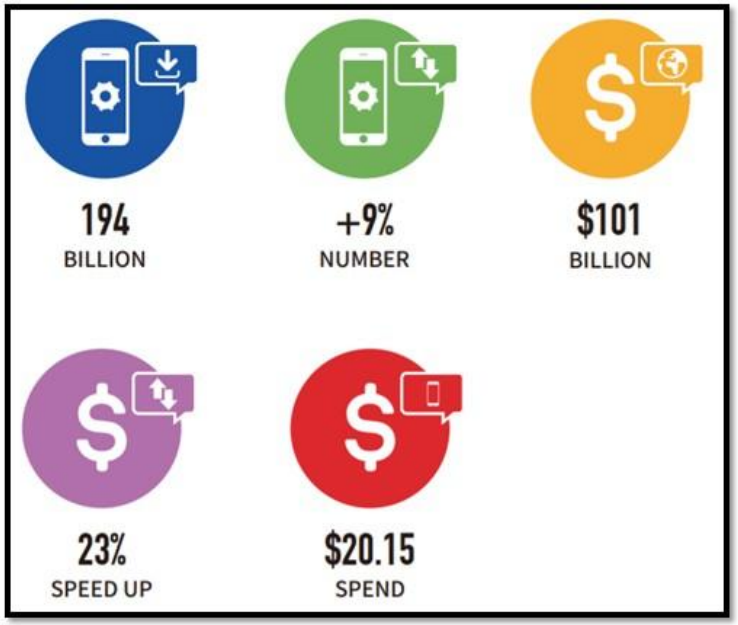
# Android Development



# Android App and Market

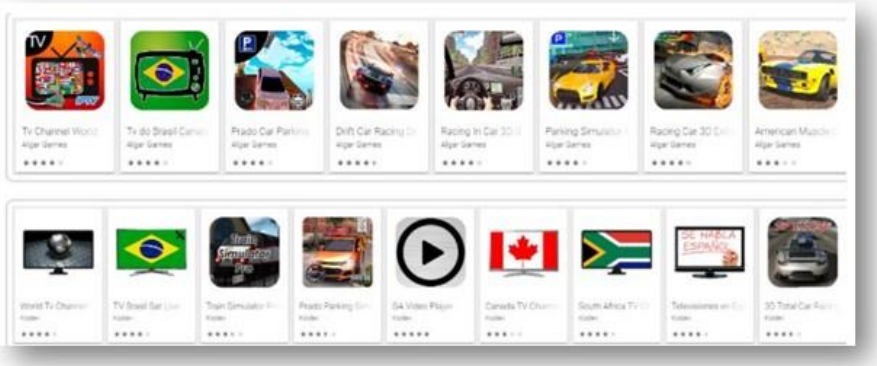


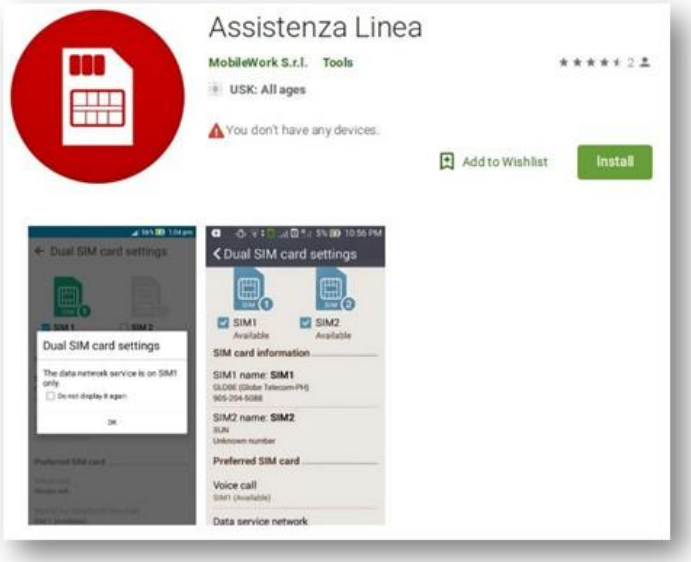
# Security Events


图：央视曝光恶意的免费充电桩

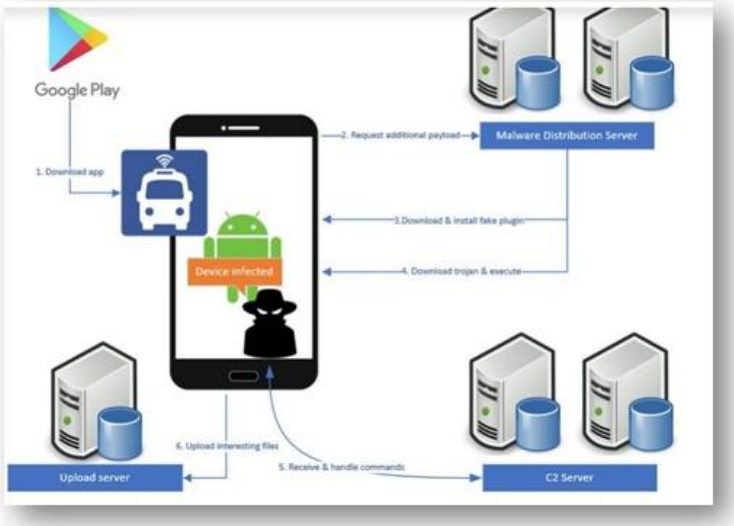


图：GooglePlay上一些广告软件虚假应用截图

# Security Events

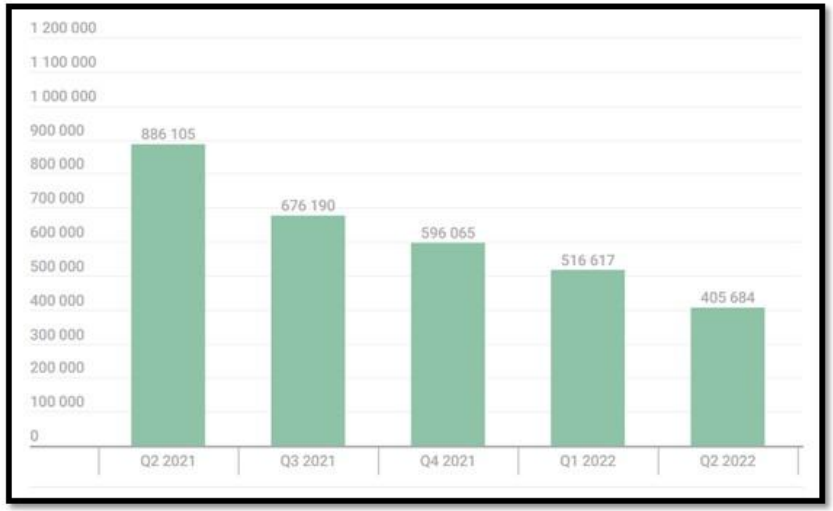


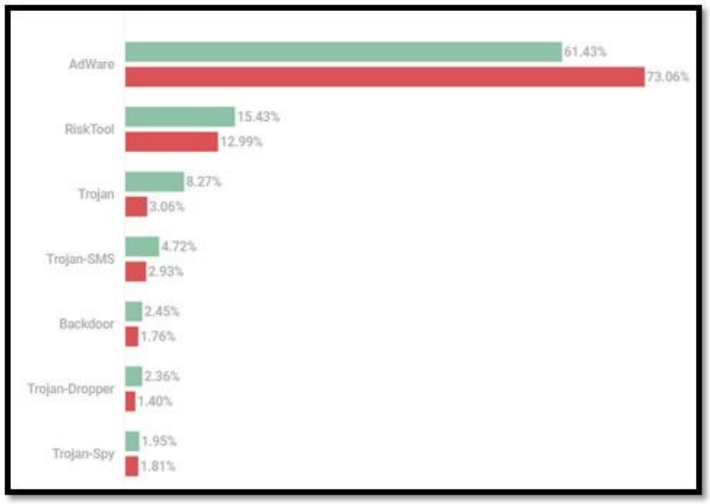

图：Exodus在Google Play中的下载页面




图：韩国公交车App感染流程

# Security Risk within Android



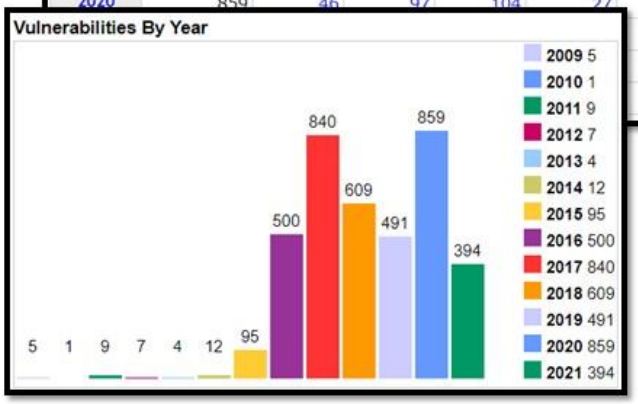



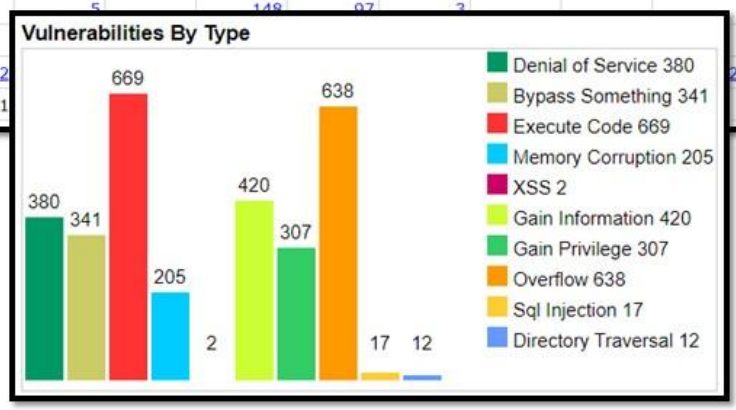
# Cost of Security Android

- 1. Malware attacks cost the average US business well over $2 million.
- 2. Five-year forecasts for malware security are set to be in the $1 trillion ballpark.
- 3. A malware attack can cost 50 days in time to defense.
- 4. Malware costs have risen by over 20% per year.
- ……
- 11. Yahoo had 3 billion accounts compromised
- 12. Reports show that 37.5 million records were stolen in one of the largest malware heists in history.

## Vulnerability Trends Over Time

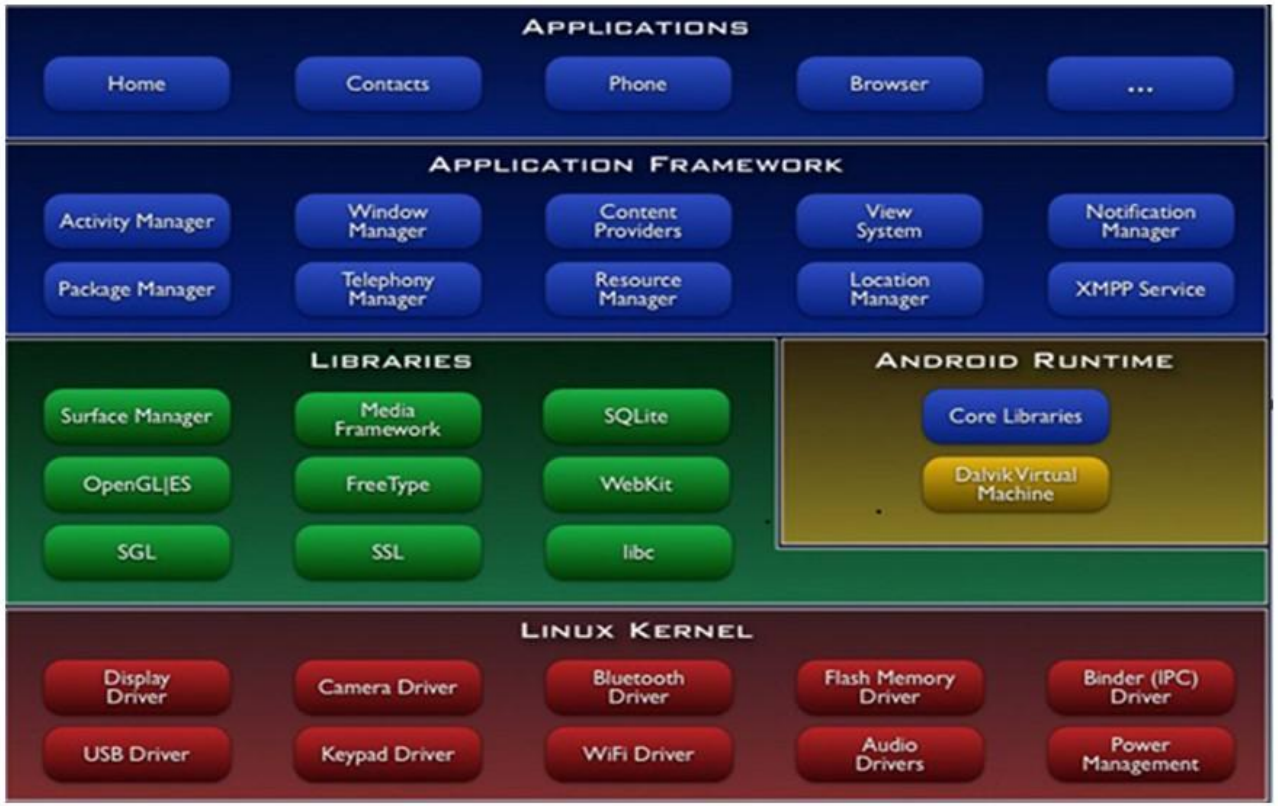
| Year | # of Vulnerabilities | DoS | Code Execution | Overflow | Memory Corruption | Sql Injection | XSS | Directory Traversal | Http Response Splitting | Bypass something | Gain Information | Gain Privileges | CSRF | File Inclusion | # of exploits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2009 | 5 | 3 | | | | | | | | 1 | | | | | |
| 2010 | 1 | 1 | 1 | | | | | | | | | | | | |
| 2011 | 9 | 1 | 1 | | 1 | | 1 | | | 4 | 1 | 3 | | | |
| 2012 | 7 | 5 | 3 | 2 | | | | | | | 1 | | | | 1 |
| 2013 | 4 | | 1 | 1 | 1 | | | | | 1 | 1 | 1 | | | |
| 2014 | 12 | 2 | 4 | 1 | | 1 | | | | 1 | 2 | 1 | | | 1 |
| 2015 | 95 | 46 | 49 | 50 | 37 | | | | | 13 | 14 | 17 | | | |
| 2016 | 500 | 104 | 72 | 91 | 38 | | | | | 47 | 96 | 236 | | | |
| 2017 | 840 | 86 | 206 | 170 | 32 | | | 1 | | 30 | 113 | 36 | | | |
| 2018 | 609 | 32 | 84 | 143 | 12 | 2 | 1 | 2 | | 17 | 63 | 3 | | | |
| 2019 | 491 | 37 | 107 | 41 | 24 | 3 | | 1 | | 39 | 22 | 1 | | | |
| 2020 | 859 | 46 | 97 | 104 | 27 | 9 | | 5 | | 148 | 97 | 3 | | | |
| | | | | | | 2 | | | | | | | | | |
| | | | | | | 17 | 2 | | | | | | | | 2 |
| | | | | | | 0.4 | 0.1 | | | | | | | | |



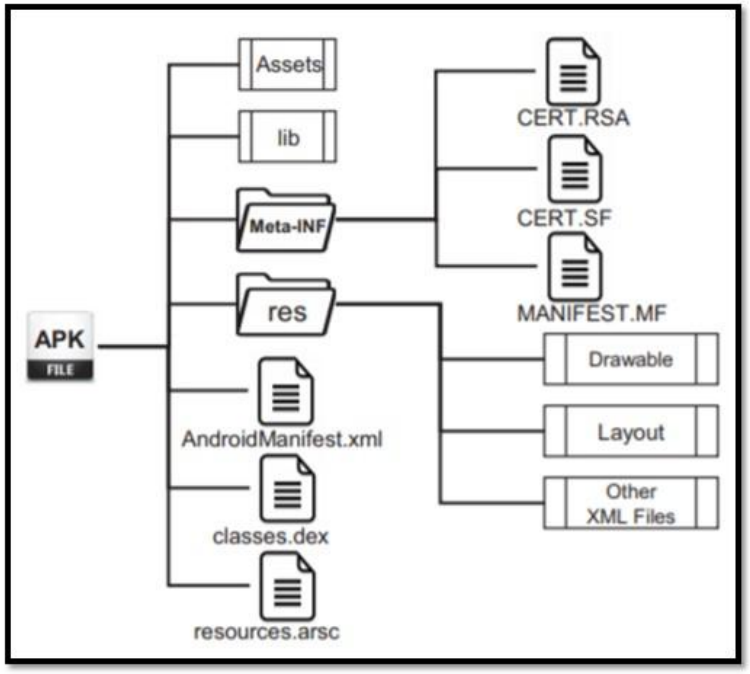



# Android Structure

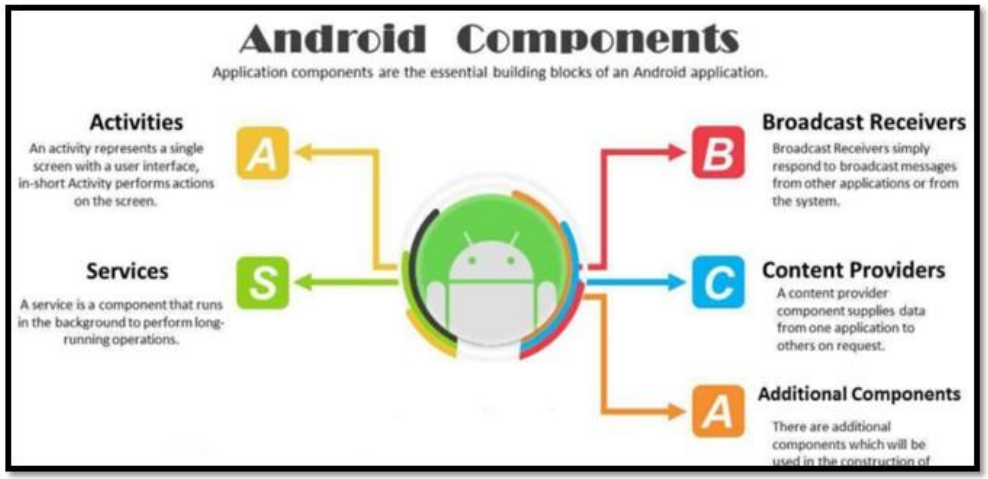
android.app -
android.content -
android.database
android.opengl
android.os -
android.text -
android.view -
android.widget -
android.webkit

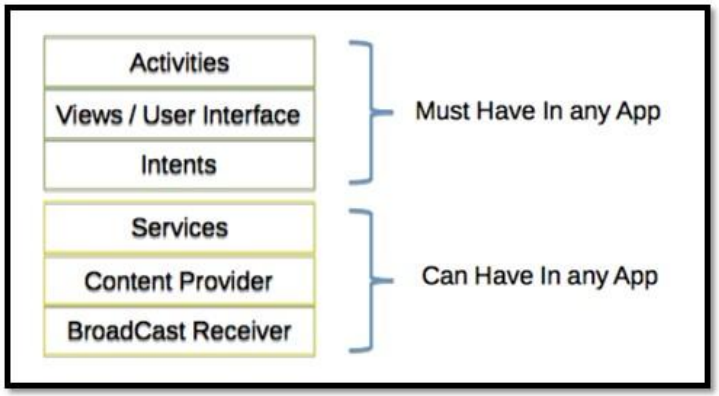


# App File Structure

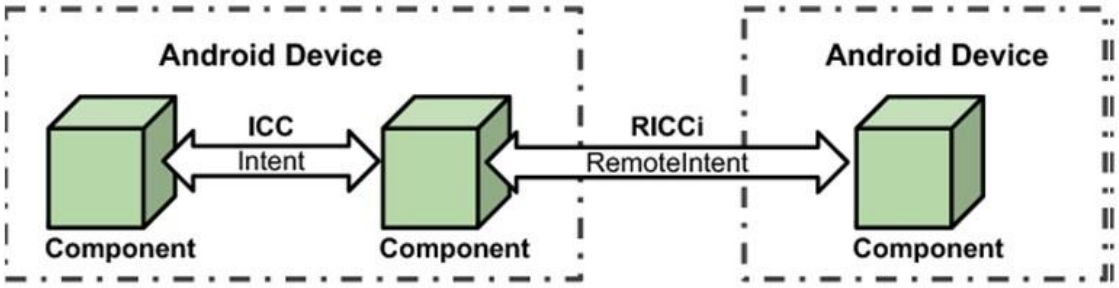


# Component


Android Components
Application components are the essential building blocks of an Android application.
Activities
An activity represents a single screen with a user interface, in-short Activity performs actions on the screen.
Services
A service is a component that runs in the background to perform long-running operations.
Broadcast Receivers
Broadcast Receivers simply respond to broadcast messages from other applications or from the system.
Content Providers
A content provider component supplies data from one application to others on request.
Additional Components
There are additional components which will be


Activities
Views / User Interface
Intents
Must Have In any App
Services
Content Provider
BroadCast Receiver
Can Have In any App

# Inter-component Communication (ICC)



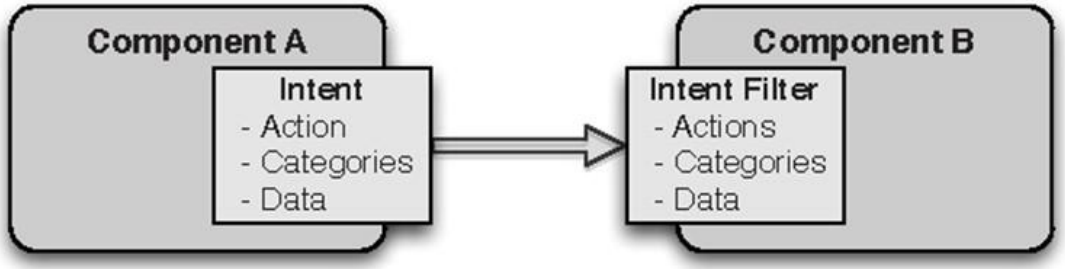
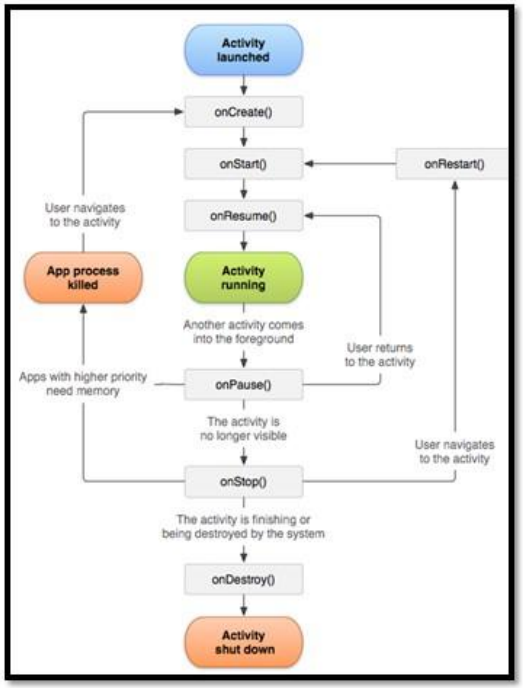
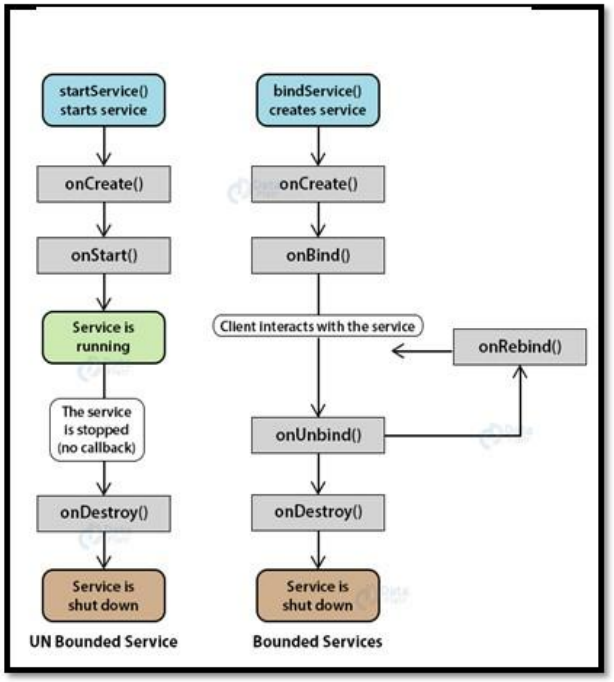



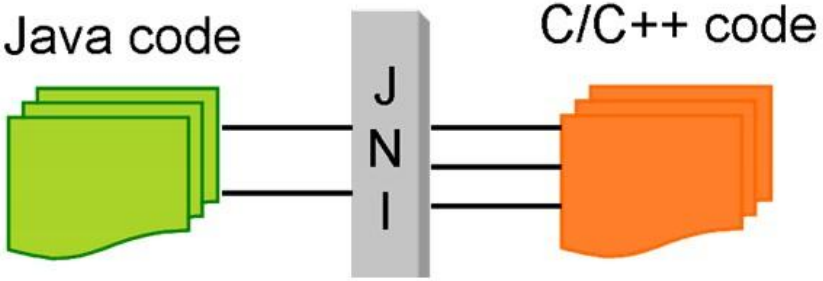

Figure 1: Implicit Intent ICC

# Android Lifecycle



# Java Native Interface (JNI)




Why?

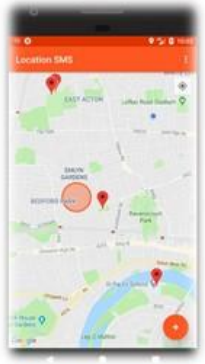
Safe & Efficiency

# Permission

Totally 324

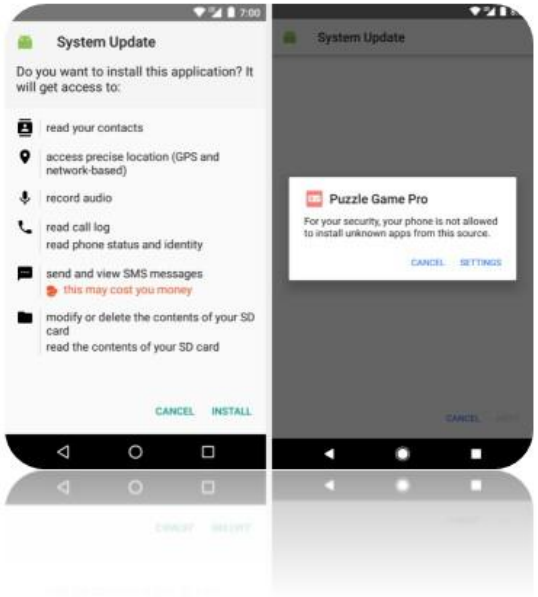
```
1   android.permission.ACCESS_ALL_DOWNLOADS
2   android.permission.ACCESS_BLUETOOTH_SHARE
3   android.permission.ACCESS_CACHE_FILESYSTEM
4   android.permission.ACCESS_CHECKIN_PROPERTIES
5   android.permission.ACCESS_CONTENT_PROVIDERS_EXTERNALLY
6   android.permission.ACCESS_DOWNLOAD_MANAGER
7   android.permission.ACCESS_DOWNLOAD_MANAGER_ADVANCED
8   android.permission.ACCESS_DRM_CERTIFICATES
9   android.permission.ACCESS_EPHEMERAL_APPS
10  android.permission.ACCESS_FM_RADIO
11  android.permission.ACCESS_INPUT_FLINGER
12  android.permission.ACCESS_KEYGUARD_SECURE_STORAGE
13  android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
14  android.permission.ACCESS_MOCK_LOCATION
15  android.permission.ACCESS_MTP
16  android.permission.ACCESS_NETWORK_CONDITIONS
17  android.permission.ACCESS_NETWORK_STATE
18  android.permission.ACCESS_NOTIFICATIONS
19  android.permission.ACCESS_NOTIFICATION_POLICY
20  android.permission.ACCESS_PDB_STATE
21  android.permission.ACCESS_SURFACE_FLINGER
22  android.permission.ACCESS_VOICE_INTERACTION_SERVICE
23  android.permission.ACCESS_VR_MANAGER
24  android.permission.ACCESS_WIFI_STATE
```

Apply for

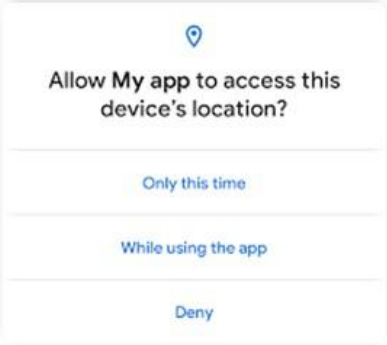




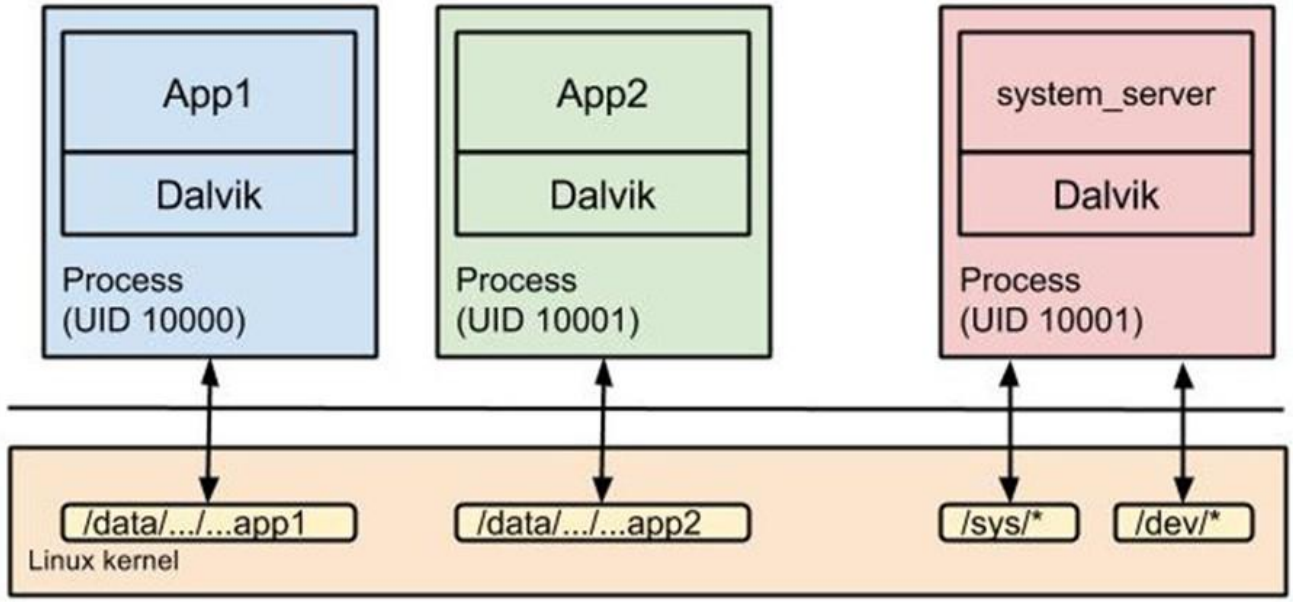
# Permission


System Update
Do you want to install this application? It will get access to:
read your contacts
access precise location (GPS and network-based)
record audio
read call log
read phone status and identity
send and view SMS messages
this may cost you money
modify or delete the contents of your SD card
read the contents of your SD card
CANCEL
INSTALL
System Update
Puzzle Game Pro
For your security, your phone is not allowed to install unknown apps from this source.
CANCEL
SETTINGS


Allow Hangouts to send and view SMS messages?
DENY
ALLOW


Allow My app to access this device's location?
Only this time
While using the app
Deny

# Android SandBoxing



# Security Problem in Android App

- Leaks (inner-component, ICC, SD card…)
- Vulnerability (DoS, Arbitrary execution, Buffer overflow…)
- Permission exploitation (Privilege escalation)
- Malware
- Repackaging
- Cryptography

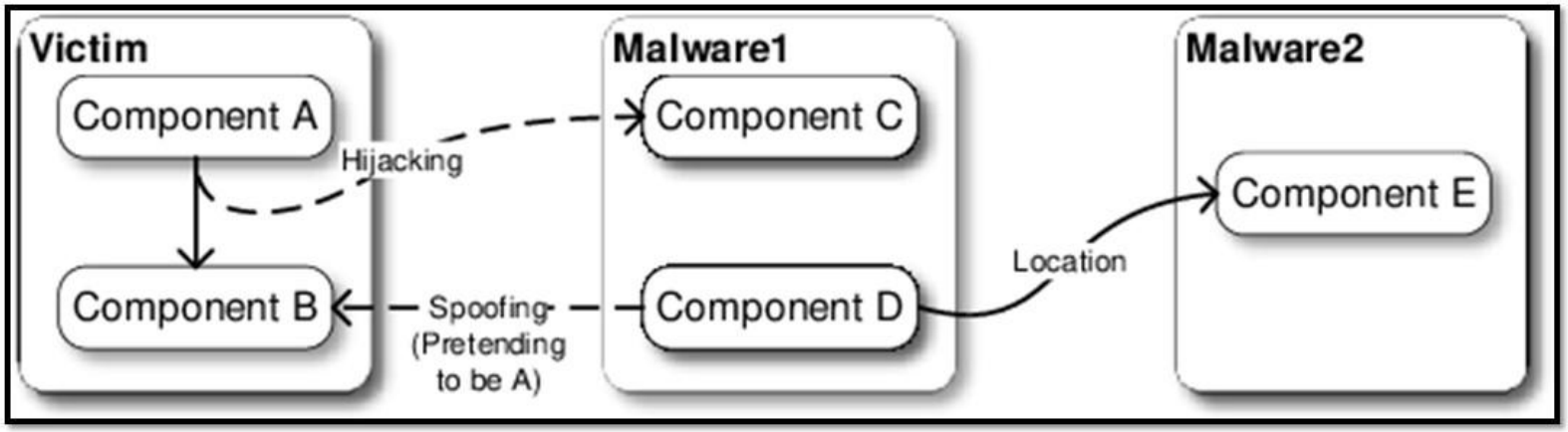
# A Case for Permission


App icon of a legitimate app is clicked by the victim.

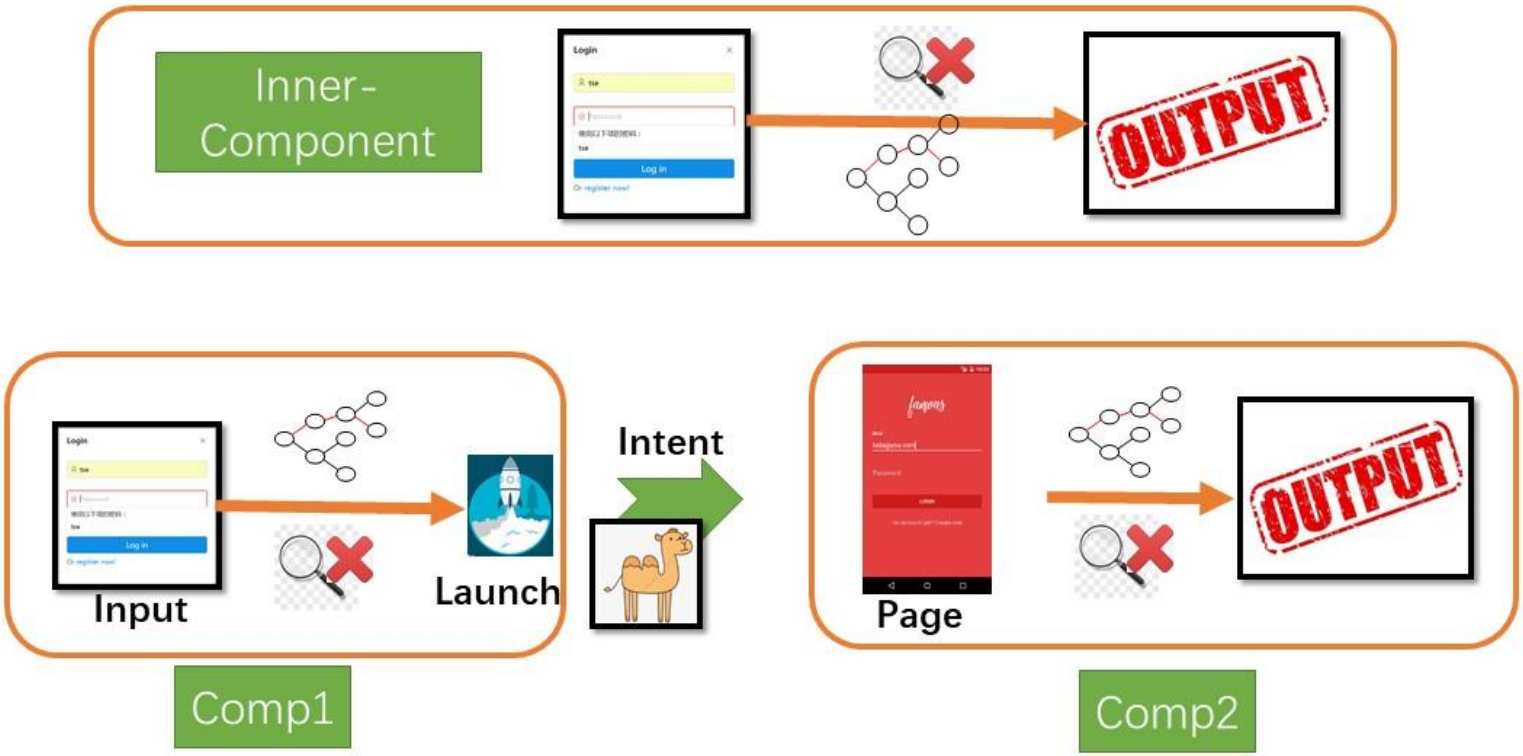
Instead of seeing the legitimate app, the malware is displayed and can now ask for any permission while pretending to be the legitimate app.

The victim is unknowingly giving permissions to the hacker. Victim is directed to the legitimate app.

# ICC-based Attack



# ICC-based Leakage

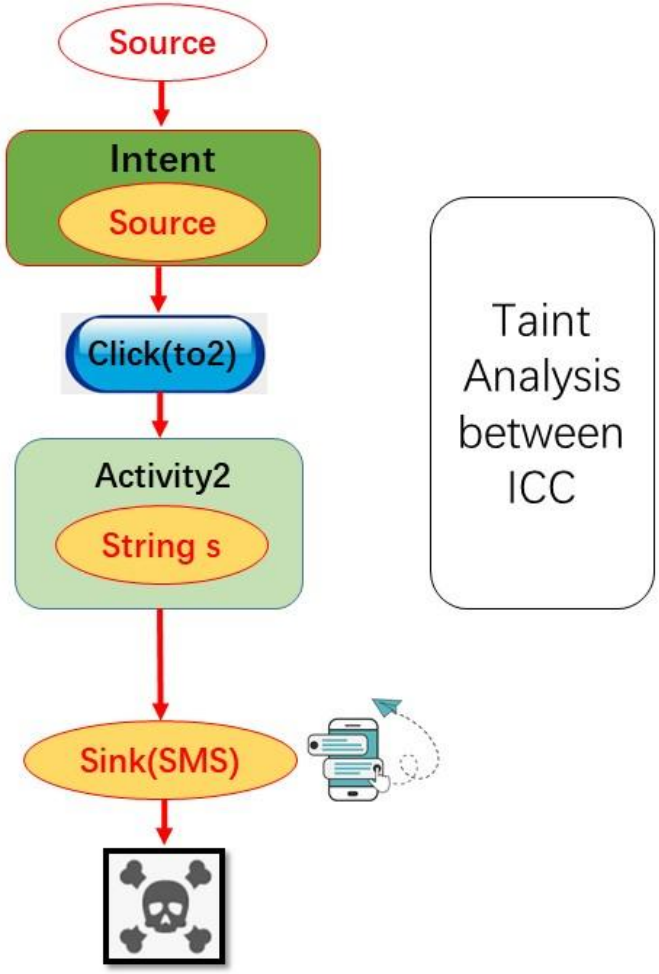


# ICC-based Leakage

```
1  //TelephonyManager telMnger; (default)
2  //SmsManager sms; (default)
3  class Activity1 extends Activity {
4   void onCreate(Bundle state) {
5    Button to2 = (Button) findViewById(to2a);
6    to2.setOnClickListener(new OnClickListener(){
7     void onClick(View v) {
8      String id = telMnger.getDeviceId();
9      Intent i = new
        Intent(Activity1.this,Activity2.class);
10     i.putExtra("sensitive", id);
11     Activity1.this.startActivity(i);
12 }});}}
13 class Activity2 extends Activity {
14  void onStart() {
15   Intent i = getIntent();
16   String s = i.getStringExtra("sensitive");
17   sms.sendTextMessage(number,null,s,null,null);
18 }}
```

Source

Sink




Taint Analysis between ICC

# Detection ICC-based Leakage

Apparecium [1], AppAudit [2], AppCaulk [3], AppIntent [4], Apposcopy [5], AppSealer [6], AsDroid [7], Bastani et al. [8], Bonett et al. [9], Brox [10], Capper [11], Cortesi et al. [12], DescribeMe [13], Dflow + DroidInfer [14], DroidJust [15], DynaLog [16], **FlowDroid [17],** Graa et al. [18], HelDroid [19], HornDroid [20], **IccTA [21],** IFT [22], Jiang et al. [23], Octeau et al. [24], Relda [25], Sufatrio et al. [26], TASMAN [27], Tuan et al. [28], Uranine [29], WeChecker [30], ATFuzzer [31], Chen et al. [32], ContentScope [33], DroidAlarm [34], **DroidSafe [35],** DroidUnPACK [36], DroidVulMon [37], Schoepe et al. [38], Tiwari [39], **AmanDroid[40], Epicc[41], RAICC[42],** CHEX[43], Didfail[44], ArrayICC[45]

# Reference 1


- [1] Titze, D. and Schütte, J., 2015. Apparecium: Revealing data flows in android applications. In 29th International Conference on Advanced Information Networking and Applications, pp. 579-586.
- [2] Xia, M., Gong, L., Lyu, Y., Qi, Z. and Liu, X., 2015. Effective real-time android application auditing. In IEEE Symposium on Security and Privacy, pp. 899-914.
- [3] Schütte, J., Titze, D. and De Fuentes, J.M., 2014. Appcaulk: Data leak prevention by injecting targeted taint tracking into android apps. In 13th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 370-379.
- [4] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P. and Wang, X.S., 2013. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 1043-1054.
- [5] Feng, Y., Anand, S., Dillig, I. and Aiken, A., 2014. Apposcopy: Semantics-based detection of android malware through static analysis. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 576-587.
- [6] Zhang, M. and Yin, H. (2014). AppSealer: Automatic Generation of Vulnerability-Specific Patches for Preventing Component Hijacking Attacks in Android Applications. In NDSS.
- [7] Huang, J., Zhang, X., Tan, L., Wang, P. and Liang, B. (2014). Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction. In Proceedings of the 36th International Conference on Software Engineering, pp. 1036-1046.
- [8] Bastani, O., Anand, S. and Aiken, A., 2015. Interactively verifying absence of explicit information flows in Android apps. ACM SIGPLAN Notices, 50(10), pp.299-315.
- [9] Bonett, R., Kafle, K., Moran, K., Nadkarni, A. and Poshyvanyk, D., 2018. Discovering flaws in security-focused static analysis tools for android using systematic mutation. In 27th {USENIX} Security Symposium ({USENIX} Security 18), pp. 1263-1280.
- [10] Ma, S., Tang, Z., Xiao, Q., Liu, J., Duong, T.T., Lin, X. and Zhu, H., 2013. Detecting GPS information leakage in Android applications. In 2013 IEEE Global Communications Conference (GLOBECOM), pp. 826-831.
- [11] Zhang, M. and Yin, H., 2014. Efficient, context-aware privacy leakage confinement for android applications without firmware modding. In Proceedings of the 9th ACM symposium on Information, computer and communications security, pp. 259-270.
- [12] Cortesi, A., Ferrara, P., Pistoia, M. and Tripp, O., 2015. Datacentric semantics for verification of privacy policy compliance by mobile applications. In International Workshop on Verification, Model Checking, and Abstract Interpretation pp. 61-79.

# Reference 2


- [13] Zhang, M., Duan, Y., Feng, Q. and Yin, H., 2015. Towards automatic generation of security-centric descriptions for android apps. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 518-529.
- [14] Huang, W., Dong, Y., Milanova, A. and Dolby, J., 2015. Scalable and precise taint analysis for android. In Proceedings of the 2015 International Symposium on Software Testing and Analysis, pp. 106-117.
- [15] Chen, X. and Zhu, S., 2015. DroidJust: automated functionality-aware privacy leakage analysis for Android applications. In Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, pp. 1-12.
- [16] Alzaylaee, M.K., Yerima, S.Y. and Sezer, S., 2016. DynaLog: An automated dynamic analysis framework for characterizing android applications. In International Conference on Cyber Security and Protection Of Digital Services (Cyber Security), pp. 1-8.
- [17] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D. and McDaniel, P., 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. Acm Sigplan Notices, 49(6), pp.259-269.
- [18] Graa, M., Cuppens-Boulahia, N., Cuppens, F., Lanet, J.L. and Moussaileb, R., 2017, May. Detection of side channel attacks based on data tainting in android systems. In IFIP International Conference on ICT Systems Security and Privacy Protection, pp. 205-218.
- [19] Andronio, N., Zanero, S. and Maggi, F., 2015. Heldroid: Dissecting and detecting mobile ransomware. In International Symposium on Recent Advances in Intrusion Detection, pp. 382-404.
- [20] Calzavara, S., Grishchenko, I. and Maffei, M., 2016. HornDroid: Practical and sound static analysis of Android applications by SMT solving. In European Symposium on Security and Privacy (EuroS&P), pp. 47-62.
- [21] Li, L., Bartel, A., Bissyandé, T.F., Klein, J., Le Traon, Y., Arzt, S., Rasthofer, S., Bodden, E., Octeau, D. and McDaniel, P., 2015. Iccta: Detecting inter-component privacy leaks in android apps. In 37th IEEE International Conference on Software Engineering, 1, pp. 280-291.
- [22] Ernst, M.D., Just, R., Millstein, S., Dietl, W., Pernsteiner, S., Roesner, F., Koscher, K., Barros, P.B., Bhoraskar, R., Han, S. and Vines, P., 2014. Collaborative verification of information flow for a high-assurance app store. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, pp. 1092-1104.
- [23] Jiang, H., Yang, H., Qin, S., Su, Z., Zhang, J. and Yan, J., 2017. Detecting energy bugs in Android apps using static analysis. In International Conference on Formal Engineering Methods, pp. 192-208.
- [24] Octeau, D., Jha, S., Dering, M., McDaniel, P., Bartel, A., Li, L., Klein, J. and Le Traon, Y., 2016. Combining static analysis with probabilistic models to enable market-scale android inter-component analysis. In Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 469-484.

# Reference 3


- [25] Guo, C., Zhang, J., Yan, J., Zhang, Z. and Zhang, Y., 2013. Characterizing and detecting resource leaks in Android applications. In 28th International Conference on Automated Software Engineering (ASE), pp. 389-398.
- [26] Chua, T.W., Tan, D.J. and Thing, V.L., 2015. Accurate specification for robust detection of malicious behavior in mobile environments. In European Symposium on Research in Computer Security, pp. 355-375.
- [27] Arzt, S., Rasthofer, S., Hahn, R. and Bodden, E., 2015. Using targeted symbolic execution for reducing false-positives in dataflow analysis. In Proceedings of the 4th ACM SIGPLAN International Workshop on State of the Art in Program Analysis, pp. 1-6.
- [28] Tuan, L.H., Cam, N.T. and Pham, V.H., 2019. Enhancing the accuracy of static analysis for detecting sensitive data leakage in Android by using dynamic analysis. Cluster Computing, 22(1), pp.1079-1085.
- [29] Al Nidawi, H.S.A., Wei, K.T., Dawood, K.A. and Khaleel, A., 2017. Energy Consumption Patterns of Mobile Applications in Android Platform: A Systematic Literature Review. Journal of Theoretical & Applied Information Technology, 95(24).
- [30] Cui, X., Wang, J., Hui, L.C.., 2015. Wechecker: efficient and precise detection of privilege escalation vulnerabilities in android apps. In Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, pp. 1-12.
- [31] Karim, I., Cicala, F., Hussain, S.R., 2019. Opening Pandora's box through ATFuzzer: dynamic analysis of AT interface for Android smartphones. In Proceedings of the 35th Computer Security Applications Conference, pp. 529-543.
- [32] Chen, H., Leung, H.F., Han, B. and Su, J., 2017. Automatic privacy leakage detection for massive android apps via a novel hybrid approach. In International Conference on Communications (ICC), pp. 1-7.
- [33] Jiang, Y.Z.X. and Xuxian, Z., 2013. Detecting passive content leaks and pollution in android applications. In Proceedings of the 20th Network and Distributed System Security Symposium (NDSS).
- [34] Zhongyang, Y., Xin, Z., Mao, B. and Xie, L., 2013. DroidAlarm: an all-sided static analysis tool for Android privilegeescalation malware. In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 353-358.
- [35] Gordon, M.I., Kim, D., Perkins, J.H., Gilham, L., Nguyen, N. and Rinard, M.C., 2015. Information flow analysis of android applications in droidsafe. In NDSS, 15 (201), p. 110.
- [36] Duan, Y., Zhang, M., Bhaskar, A.V., Yin, H., Pan, X., Li, T., Wang, X. and Wang, X., 2018. Things You May Not Know About Android (Un) Packers: A Systematic Study based on Whole-System Emulation. In NDSS.

# Reference 4


- [37] Ham, Y.J., Lee, H.W., Lim, J.D. and Kim, J.N., 2013. DroidVulMon--Android Based Mobile Device Vulnerability Analysis and Monitoring System. In Seventh International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 26-31.
- [38] Schoepe, D., Balliu, M., Piessens, F. and Sabelfeld, A., 2016. Let's face it: Faceted values for taint tracking. In European Symposium on Research in Computer Security, pp. 561-580.
- [39] Tiwari, A., 2019. Enhancing Users' Privacy: Static Resolution of the Dynamic Properties of Android (University of Potsdam).
- [40] Wei F, Roy S, Ou X. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps[C]//Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014: 1329-1341.
- [41] Octeau D, McDaniel P, Jha S, et al. Effective inter-component communication mapping in android: An essential step towards holistic security analysis[C]//22nd {USENIX} Security Symposium ({USENIX} Security 13). 2013: 543-558.
- [42] Samhi J, Bartel A, Bissyandé T F, et al. RAICC: Revealing Atypical Inter-Component Communication in Android Apps[C]//2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021: 1398-1409.
- [43] Lu L, Li Z, Wu Z, et al. Chex: statically vetting android apps for component hijacking vulnerabilities[C]//Proceedings of the 2012 ACM conference on Computer and communications security. 2012: 229-240.
- [44] Klieber W, Flynn L, Bhosale A, et al. Android taint flow analysis for app sets[C]//Proceedings of the 3rd ACM SIGPLAN International Workshop on the State of the Art in Java Program Analysis. 2014: 1-6.
- [45] Maalouf A, Lu L. Taint analysis of arrays in Android applications[C]//Proceedings of the 36th Annual ACM Symposium on Applied Computing. 2021: 893-899.

# SuSi

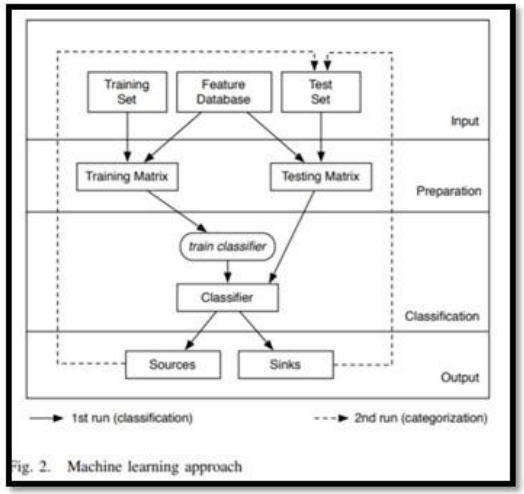


Fig. 2. Machine learning approach

SVM Classifier

| Category | Recall [%] | Precision [%] |
|---|---|---|
| ACCOUNT | 100.0 | 100.0 |
| BLUETOOTH | 83.3 | 100.0 |
| BROWSER | 83.0 | 100.0 |
| CALENDAR | 100.0 | 100.0 |
| CONTACT | 95.0 | 100.0 |
| DATABASE | 50.0 | 100.0 |
| FILE | 75.0 | 100.0 |
| NETWORK | 83.3 | 83.3 |
| NFC | 100.0 | 100.0 |
| SETTINGS | 75.0 | 85.7 |
| SYNC | 100.0 | 100.0 |
| UNIQUE_IDENTIFIER | 88.9 | 100.0 |
| NO_CATEGORY | 95.7 | 62.9 |
| Weighted Average | 88.7 | 89.6 |

TABLE VI. SOURCE CATEGORY CROSS VALIDATION

| Category | Recall [%] | Precision [%] |
|---|---|---|
| ACCOUNT | 85.7 | 100.0 |
| AUDIO | 100.0 | 100.0 |
| BROWSER | 50.0 | 100.0 |
| CALENDAR | 100.0 | 100.0 |
| CONTACT | 91.7 | 100.0 |
| FILE | 60.0 | 100.0 |
| LOG | 100.0 | 71.4 |
| NETWORK | 72.7 | 88.9 |
| NFC | 100.0 | 100.0 |
| PHONE_CONNECTION | 75.0 | 85.7 |
| PHONE_STATE | 100.0 | 100.0 |
| SMS_MMS | 96.3 | 100.0 |
| SYNC | 80.0 | 100.0 |
| SYSTEM | 80.6 | 89.3 |
| VOIP | 66.7 | 100.0 |
| NO_CATEGORY | 97.1 | 70.2 |
| Weighted Average | 85.7 | 88.0 |

TABLE VII. SINK CATEGORY CROSS VALIDATION

S. Rasthofer, S. Arzt, and E. Bodden, “A machine-learning approach for classifying and categorizing android sources and sinks,” NDSS, 2014.

# Epicc

```
1 private OnClickListener mMyListener =
2     new OnClickListener() {
3   public void onClick(View v) {
4     Intent intent = new Intent();
5     intent.setAction("a.b.ACTION");
6     intent.addCategory("a.b.CATEGORY");
7     startActivity(intent); } };
```

Figure 2: Example of implicit Intent communication

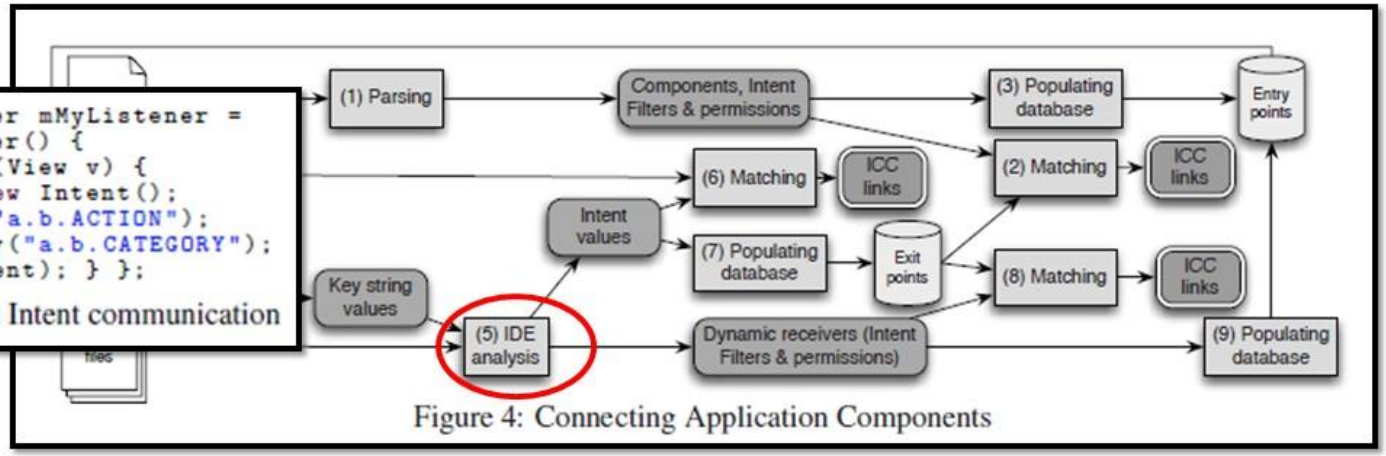


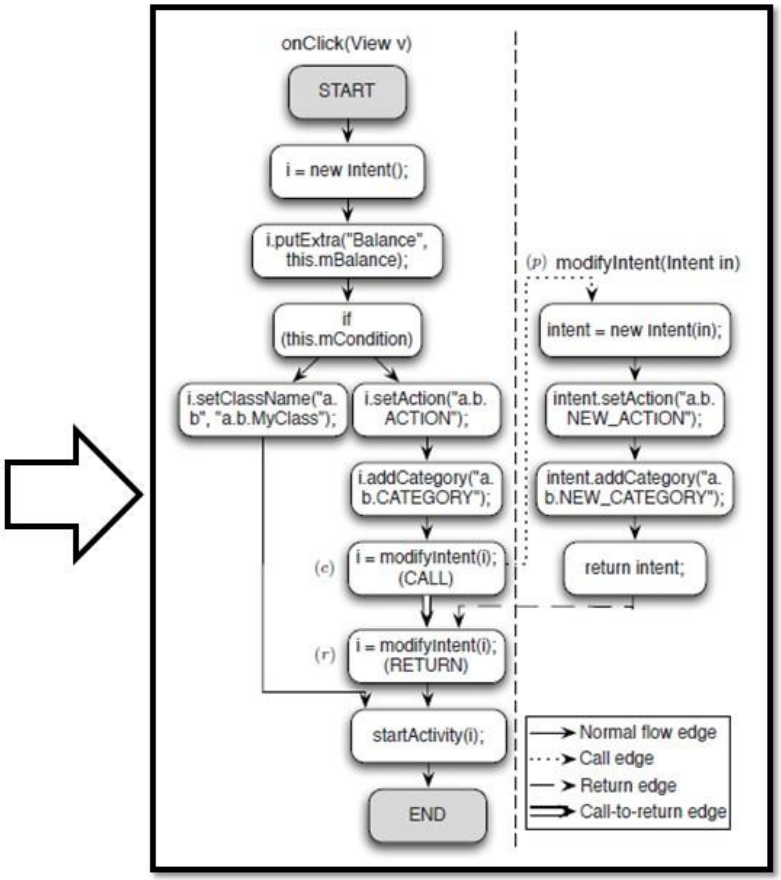

Figure 4: Connecting Application Components

IDE： Inter-procedural Distributive Environment

Octeau D, McDaniel P, Jha S, et al. Effective inter-component communication mapping in android: An essential step towards holistic security analysis[C]//22nd {USENIX} Security Symposium ({USENIX} Security 13). 2013: 543-558.

# Epicc

```
1  public void onClick(View v) {
2    Intent i = new Intent();
3    i.putExtra("Balance", this.mBalance);
4    if (this.mCondition) {
5      i.setClassName("a.b",
           "a.b.MyClass");
6    } else {
7      i.setAction("a.b.ACTION");
8      i.addCategory("a.b.CATEGORY");
9      i = modifyIntent(i);
10   }
11   startActivity(i); }
12
13 public Intent modifyIntent(Intent in) {
14   Intent intent = new Intent(in);
15   intent.setAction("a.b.NEW_ACTION");
16   intent.addCategory("a.b.NEW_CATEGORY");
17   return intent; }
```



# FlowDroid

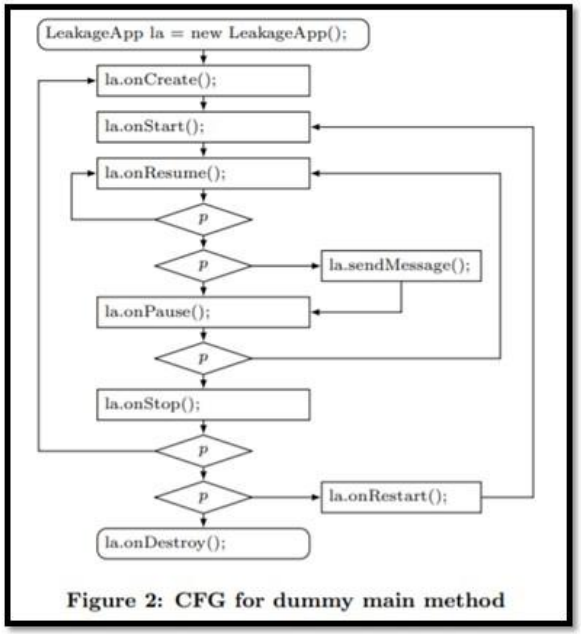



**Figure 2: CFG for dummy main method**

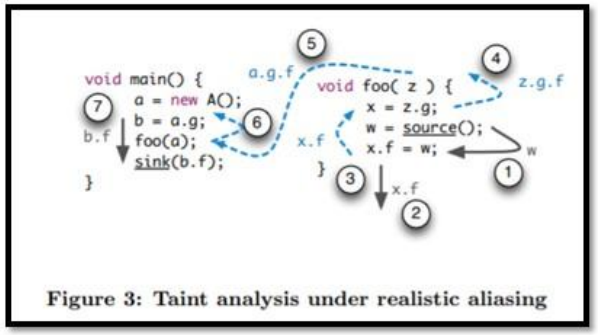



**Figure 3: Taint analysis under realistic aliasing**

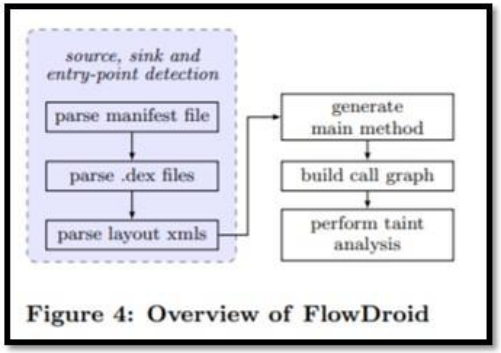



**Figure 4: Overview of FlowDroid**

Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D. and McDaniel, P., 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. Acm Sigplan Notices, 49(6), pp.259-269.

# AmanDroid

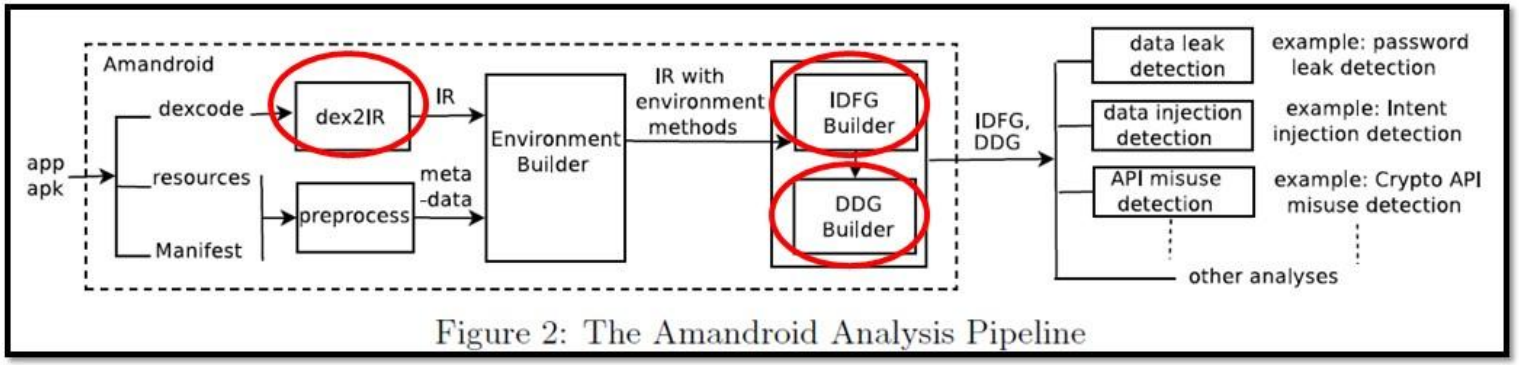



Figure 2: The Amandroid Analysis Pipeline

- IDFG: inter-component data-flow graph
- DDG： data dependence graph

Wei F, Roy S, Ou X. Amandroid: A precise and general inter-component data flow analysis framework for security vetting of android apps[C]//Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. 2014: 1329-1341.

# IccTA

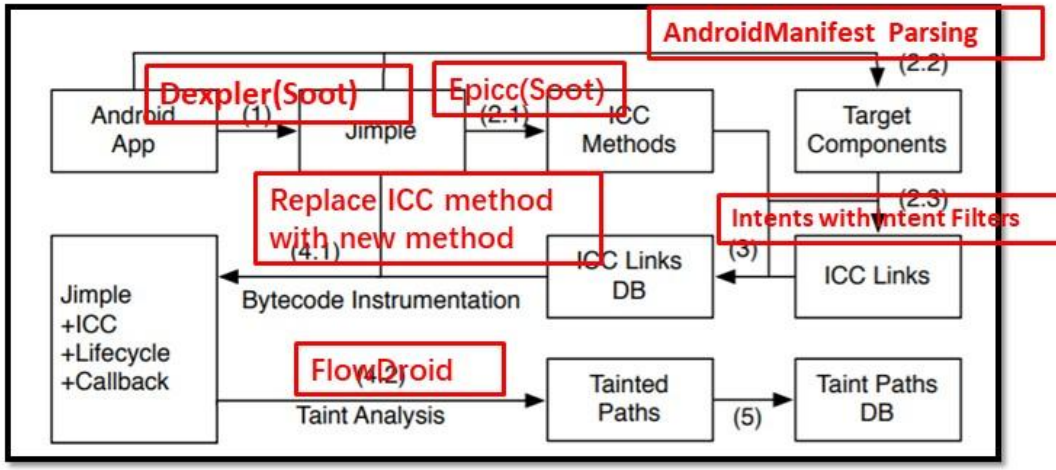
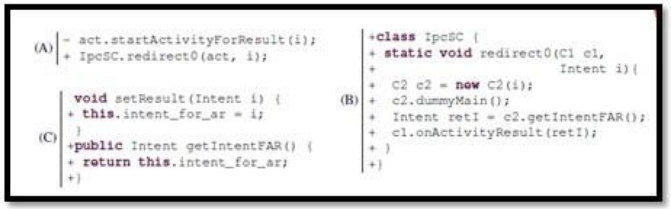



TABLE I
THE TOP 8 USED ICC METHODS. WHEN THESE METHODS ARE OVERLOADED, WE CONSIDER THE ONE WITH MOST NUMBER OF CALLS.

| ICC Method | # of Calls | # of Apps |
|---|---|---|
| startActivity | 54,334 (56.01%) | 1,972 (96.4%) |
| startActivityForResult | 11,118 (11.46%) | 1,409 (68.7%) |
| query | 8,191 (8.44%) | 1,374 (67.2%) |
| startService | 6,660 (6.86%) | 1,597 (78.1%) |
| sendBroadcast | 5,119 (5.28%) | 1,035 (50.1%) |
| insert | 2,164 (2.23%) | 780 (38.1%) |
| bindService | 1,638 (1.69%) | 512 (25.0%) |
| delete | 1,633 (1.69%) | 472 (23.1%) |
| Other ICC Methods | 6,155 (6.34%) | - |
| Total | 97,012 (100%) | - |




Li, L., Bartel, A., Bissyandé, T.F., Klein, J., Le Traon, Y., Arzt, S., Rasthofer, S., Bodden, E., Octeau, D. and McDaniel, P., 2015. Iccta: Detecting inter-component privacy leaks in android apps. In 37th IEEE International Conference on Software Engineering, 1, pp. 280-291.

# DroidSafe

Android Device Implementation (ADI)

```
1  package android.os;
2  public class Bundle ... {
3    private Map<String,Object> mMap =
4                  new HashMap<String,Object>(); (H)
5
6    public void put(String k, Object v) {
7      mMap.put(k,v);
8    }
9
10   public Object get(String k) {
11     return mMap.get(k);
12   }
13 }
```

```
1  package java.util;
2  public class HashMap<K,V>... {
3    private Entry[] table = new Entry[size]; (T)
4
5    public void put(K key, V value) {
6      ...
7      table[index] = new Entry<K,V>(key, value); (E)
8    }
9
10   public V get(Object key) {
11     ...
12     e = table[indexFor(hash, table.length)];
13     ...
14     return e;
15   }
16 }
```

Android Application Source Code

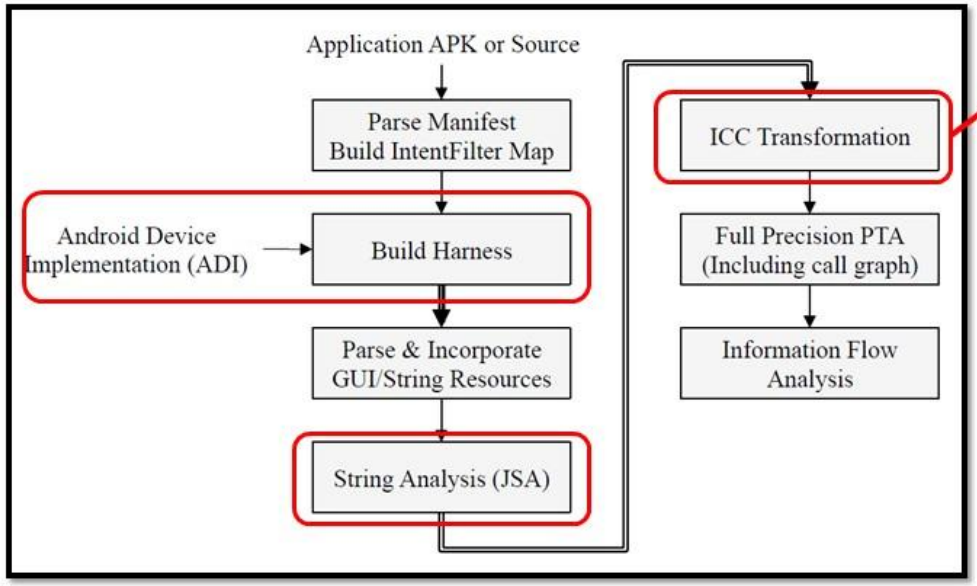
```
1  public class Activity1 extends Activity {
2    ...
3    Bundle bundle1 = new Bundle(); (N)
4    bundle1.put("data", <notSensitive>);
5    ...
6    sink(bundle1);  //not a sensitive flow
7  }
```

```
1  public class Activity2 extends Activity {
2    ...
3    double sensitive = location.getLatitude(); //source
4    Bundle bundle2 = new Bundle(); (S)
5    bundle2.put("data", sensitive);
6    ...
7    sink(bundle2);  //flow of sensitive -> sink
8  }
```

Fig. 2. Example source code for our ADI and two `Activity` objects illustrating the challenges of points-to and information flow analysis.

Gordon, M.I., Kim, D., Perkins, J.H., Gilham, L., Nguyen, N. and Rinard, M.C., 2015. Information flow analysis of android applications in droidsafe. In NDSS, 15 (201), p. 110.

# DroidSafe




| Source Method | Target Method Call Injected |
|---|---|
| Context: void send+Broadcast(Intent, ...) [6 variants] | BroadcastReceiver: void onReceive(Intent) |
| Activity: void startActivit+(Intent, ...) [6 variants] | Activity: void setIntent(Intent) |
| Context: void bindService(Intent, Connection) | Service: void droidSafeOnBind(Intent, Connection) |
| Context: void startService(Intent) | Service: void onStartCommant(Intent, ...) |
| ContentResolver: insert, query, delete, update | ContentProvider: insert, query, delete, update |

Fig. 3. DroidSafe's ICC source to target methods transformations.

- **Accurate analysis stubs for 3,176 native methods, using hook technique**
- **Inject taint on method's arguments and return value for parsing Reflection**

Gordon, M.I., Kim, D., Perkins, J.H., Gilham, L., Nguyen, N. and Rinard, M.C., 2015. Information flow analysis of android applications in Droidsafe. In NDSS, 15 (201), p. 110.

# RAICC（atypical ICC）

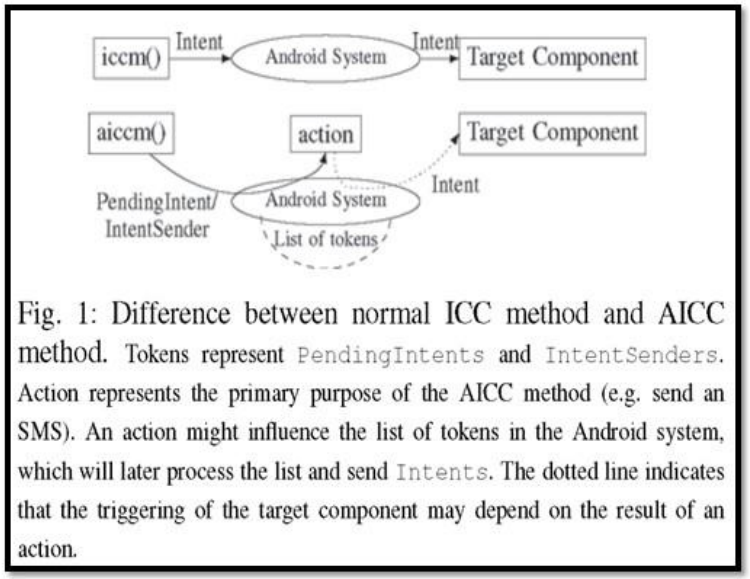



Fig. 1: Difference between normal ICC method and AICC method. Tokens represent `PendingIntents` and `IntentSenders`. Action represents the primary purpose of the AICC method (e.g. send an SMS). An action might influence the list of tokens in the Android system, which will later process the list and send `Intents`. The dotted line indicates that the triggering of the target component may depend on the result of an action.

```
1  public void reconnectionAttempts() {
2    Calendar cal = Calendar.getInstance();
3    cal.add(12, this.elapsedTime);
4    Intent i = new Intent(this, AlarmListener.class);
5    intent.putExtra("alarm_message", "Wake up Dude !");
6    PendingIntent pi = PendingIntent.getBroadcast(this,
     ↪  0, i, 134217728);
7    AlarmManager am = (AlarmManager)
     ↪  getSystemService("alarm");
8    am.set(0, cal.getTimeInMillis(), pi);
9  }
```

Listing 2: A simplified example of how the method `set` of the `AlarmManager` class is used in a malware.

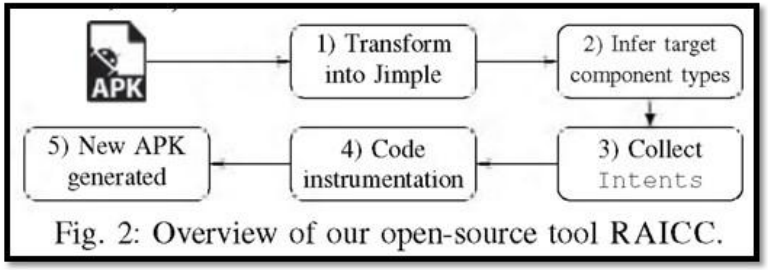



Fig. 2: Overview of our open-source tool RAICC.

Samhi J, Bartel A, Bissyandé T F, et al. RAICC: Revealing Atypical Inter-Component Communication in Android Apps[C]//2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021: 1398-1409.

# Analysis Tools

- Soot： Jimple；

https://github.com/soot-oss/soot

AndroGaurd： Smali；

https://github.com/androguard/androguard

- Gator： http://web.cse.ohio-state.edu/presto/software/gator/

# The End

guochenkai@nankai.edu.cn
过辰楷