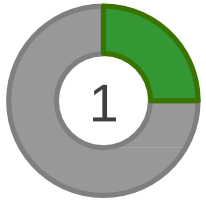
Malton: Towards On-Device Non-Invasive Mobile Malware Analysis for ART

DZ1933008 郭迎港　　MF1933040 贾晓玉

MG1933060 王文　　MG1933088 袁佳莉

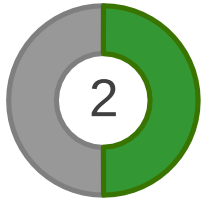
*Xue L, Zhou Y, Chen T, et al. Malton: Towards On-Device Non-Invasive Mobile Malware Analysis for {ART}[C]//26th {USENIX} Security Symposium ({USENIX} Security 17). 2017: 289-306. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/xue

Existing Android Malware Analysis Tools



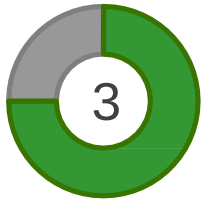
Focus on a specific layer

e.g., DroidBox (Android framework layer), DroidTrace (System layer)



Run in the emulator

e.g., DroidScope, Copperdroid (QEMU)



Modify the DVM or the compiler of ART

e.g., TaindDroid (DVM), TaintART, ARTist (dex2oat of ART)

Modify the target apps

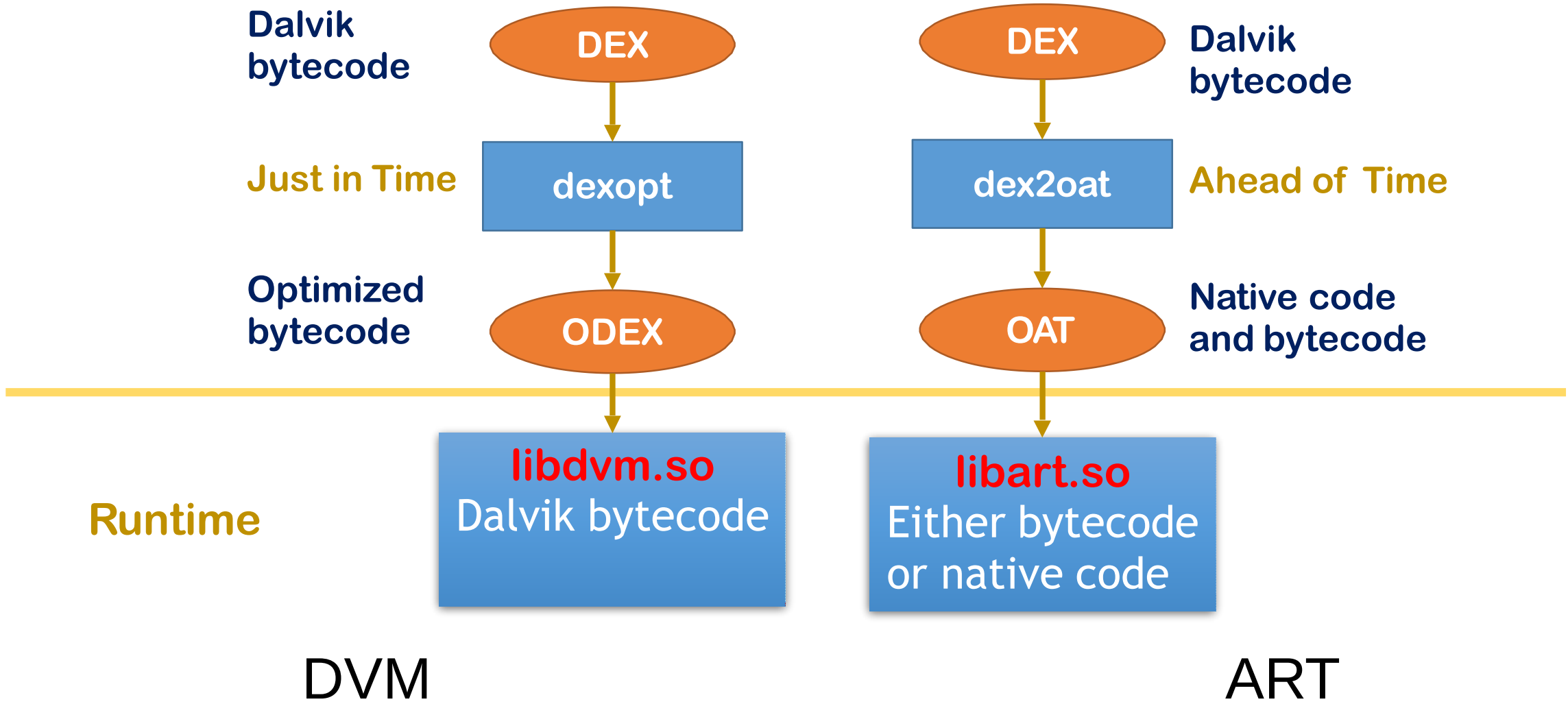
e.g., Aurasium

Malware

Mechanisms for evading detection

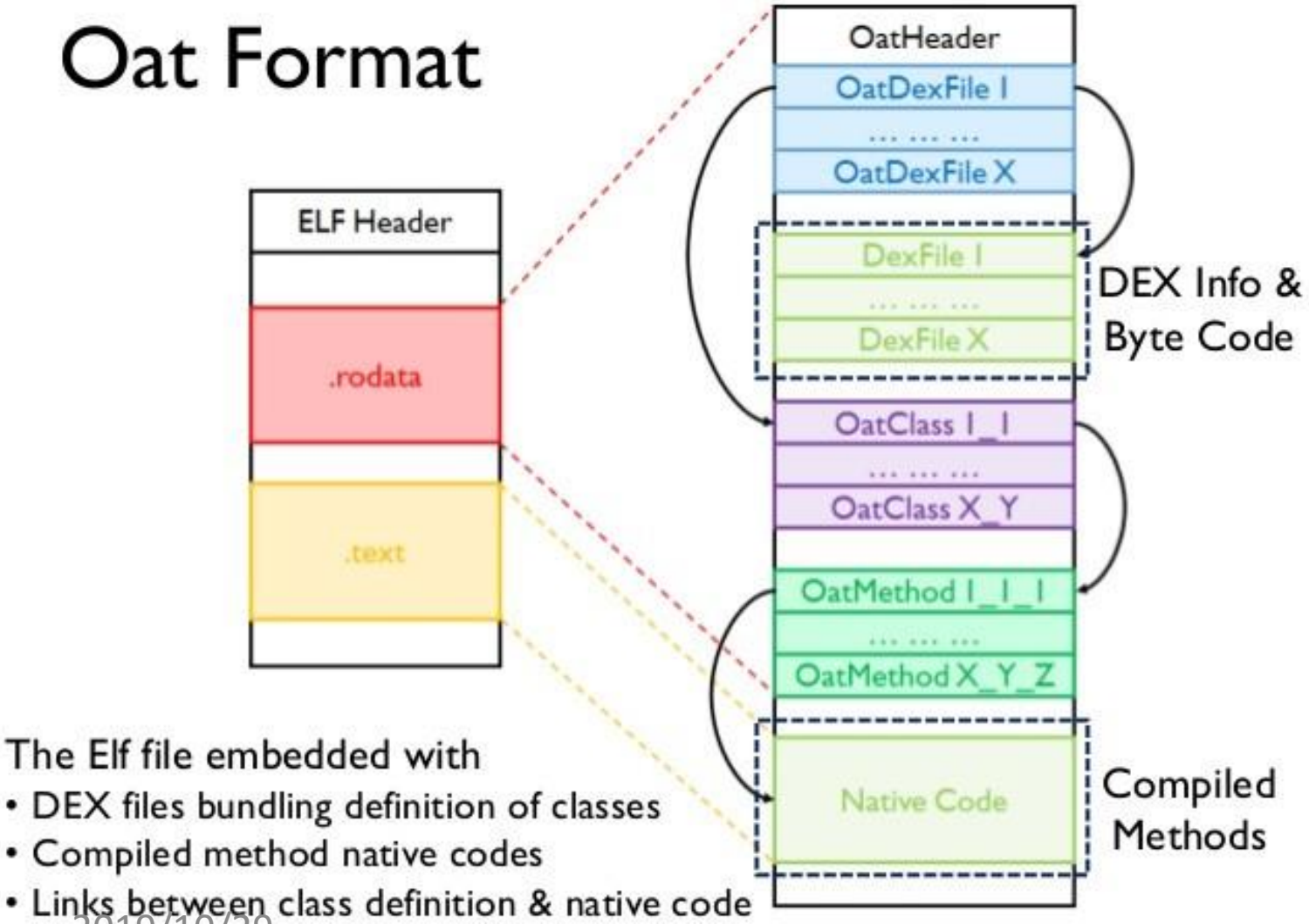
1. Detect Android emulator
2. Anti Debugging
3. Obfuscation and packing

Malton On-device and Non-invasive Analysis for ART



Android Runtime

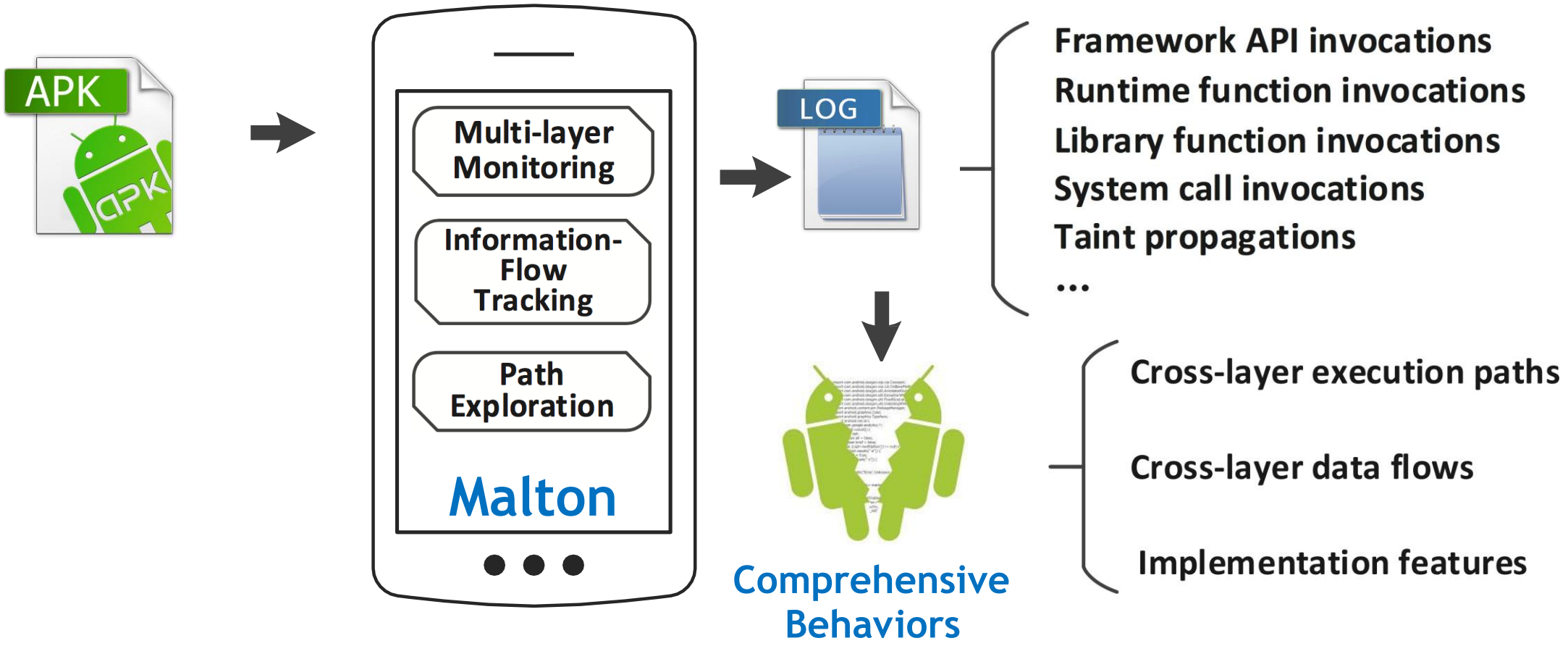
The OAT File



Parse OAT files

Get the code regions of compiled methods

Track methods according to the execution of the compiled code



- Running on a real device;
- Conducting cross-layer monitoring and information flow tracking;
- Doesn't need to modify the app.

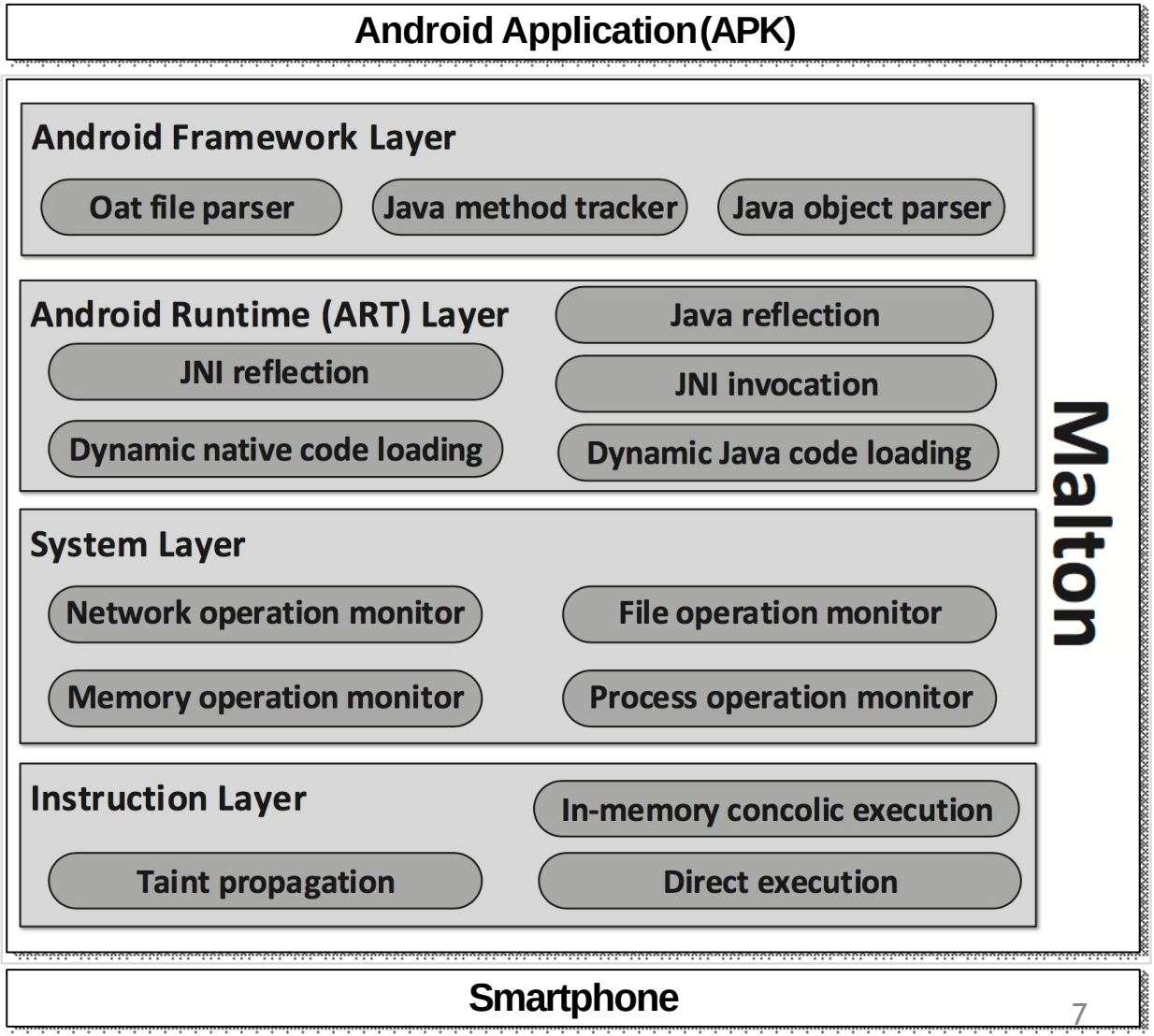
The Design of Malton

Android Framework

Android Runtime

System Libraries

Kernel



Android Framework Layer

- **OAT File Parser**
  - Get the beginning addresses of the compiled methods.
  - Get the types of the methods' parameters and return values.
- **Java Method Tracker**
  - Get information about the invoked methods with their parameters.
  - Get information about the returned methods with results.
- **Java Object Parser**
  - Get the content stored in Java class instance.
    (i.e., result of *TelephonyManager.getDeviceId()*)

Android Runtime Layer

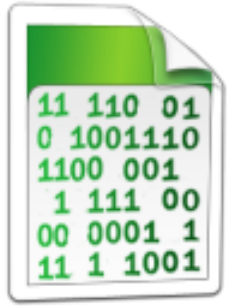
- **Native Code Loading**
  - Track the dynamically loaded native code
- **Java Code Loading**
  - Track the dynamically loaded Java code
- **JNI Invocation**
  - Track the native methods invoked by Java code
- **JNI Reflection**
  - Track the Java methods invoked by native code
- **Java Reflection**
  - Track the Java methods invoked through Java reflection

Malton can be easily extended to support the tracking of new behaviors.

System Layer

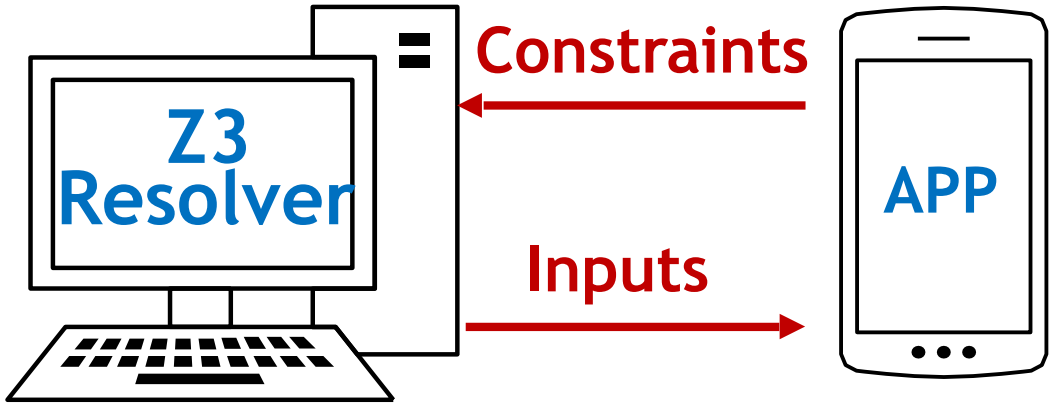
- **Network Operations**
  - Monitor information leaked through network.
  - Monitor the received remote commands.
- **File Operations**
  - Monitor the access of personal files in storage.
  - Monitor dynamically released and deleted data.
- **Memory Operations**
  - Monitor dynamically memory modification .
- **Process Operations**
  - Monitor protection behaviors (e.g., anti-emulator and anti-debugging)

Instruction Layer



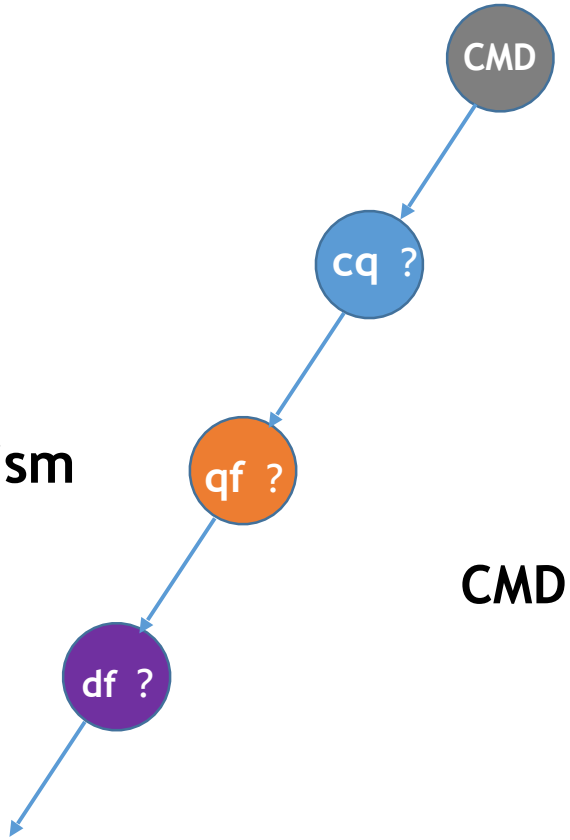
- **Taint Propagation**
  - Track the information leakage flow.
- **In-memory Concolic Execution**
  - Explore more execution paths.
- **Direct Execution**
  - Explore execution path, of which no input is generated.

Path Exploration

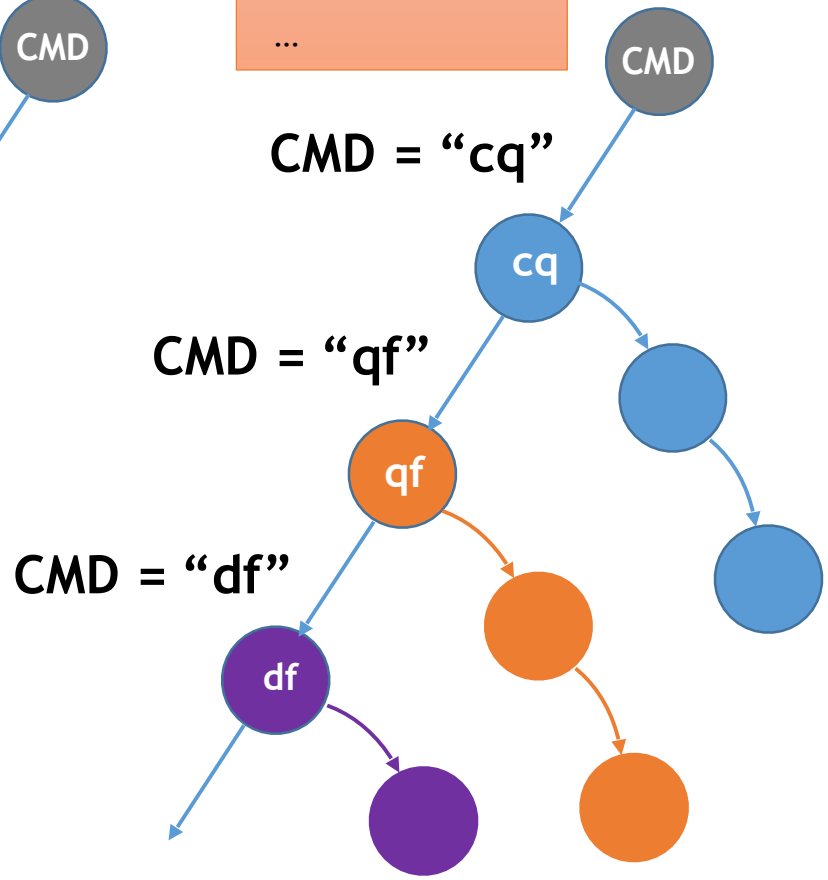


Concolic Execution with Offloading Mechanism

```
Constraints:
  CMD == "cq" ?
  CMD == "qf" ?
  CMD == "df" ?
  …
```




```
Inputs:
  CMD = "cq"
  CMD = "qf"
  CMD = "df"
  …
```



Path Exploration

Command	Detected behavior	No. of executed blocks
“cq”	Read information SMS contents, contacts, device model and system version, then send to 292019159c@fcvh77f.com with password “aAaccvv11” through SMTP protocol.	32k/20443k
“qf”	Send SMS to all contacts with no SMS content.	7k/20537k
“df”	Send SMS to specified number, and both the number and content are specified by the command SMS.	5k/22970k
“zy”	Set unconditional call forwarding through making call to “**21* targetNum%23”.	8k/22848k
“by”	Set call forwarding when the phone is busy through making call to “%23%23targetNum%23”.	15k/20639k
“ld”, “fd”, “dh”, “cz”, “fx”, “sx”, “dc”, “bc”	Modify the its configuration file zzxx.xml.	5k-18k/20403k-20452k
Others	Tell the controller the command format is error by replying an SMS.	15k/20443k

Discovering Sensitive Operations

Behavior	CopperDroid	DroidBox	Malton
Personal Info	435 (85.0%)	135 (26.4%)	511 (99.8%)
Network access	351 (68.5%)	211 (41.2%)	445 (86.9%)
File access	438 (85.5%)	509 (99.4%)	512 (100%)
Phone call	52 (10.1%)	1 (0.2%)	59 (11.5%)
Send SMS	26 (5.1%)	15 (2.9%)	28 (5.5%)
Java code loading	NA	509 (99.4%)	512 (100%)
Anti-debugging	4 (0.8%)	NA	4 (0.8%)
Native code loading	NA	NA	160 (31.2%)

- 512 samples and results of CopperDroid are downloaded from its web servers.

Table 7: Comparison of `Malton` with the popular existing Android malware analysis tools.

Tool	On device	Non-invasive	Support ART	Cross-layer Monitoring	Multi-path analysis	In-memory mechanism	Offload mechanism	Direct execution	Without modifying OS	Type
TaintDroid [39]	✓	✓	×	◔	×	×	×	×	×	Dynamic
TaintART [71]	✓	×	✓	◔	×	×	×	×	×	Dynamic
ARTist [21]	✓	×	✓	◔	×	×	×	×	×	Dynamic
DroidBox [34]	✓	✓	×	◔	×	×	×	×	×	Dynamic
VetDroid [89]	✓	✓	×	◔	×	×	×	×	×	Dynamic
DroidScope [83]	×	✓	×	◔	×	×	×	×	✓	Dynamic
CopperDroid [73]	×	✓	✓	◔	×	×	×	×	✓	Dynamic
Dagger [84]	✓	✓	✓	◔	×	×	×	×	✓	Dynamic
ARTDroid [30]	✓	✓	✓	◔	×	×	×	×	✓	Dynamic
Boxify [20]	✓	✓	✓	◔	×	×	×	×	✓	Dynamic
CRePE [29]	✓	✓	×	◔	×	×	×	×	×	Dynamic
DroidTrace [91]	✓	✓	✓	◔	×	×	×	×	✓	Dynamic
DroidTrack [64]	✓	✓	×	◔	×	×	×	×	×	Dynamic
MADAM [35]	✓	✓	✓	◔	×	×	×	×	×	Dynamic
HARVESTER [61]	✓	✓	✓	◔	✓	×	×	✓	✓	Hybrid
AppAudit [79]	×	×	×	◔	×	×	×	×	×	Hybrid
GroddDroid [10]	✓	×	✓	◔	✓	×	×	✓	✓	Hybrid
ProfileDroid [76]	✓	✓	✓	◔	×	×	×	×	✓	Hybrid
`Malton`	✓	✓	✓	●	✓	✓	✓	✓	✓	Dynamic

1. Propose and develop Malton, a novel on-device non-invasive analysis tool for ART.

2. Malton can provide a comprehensive view of the Android malware behaviors through multi-layer tracking and path exploration.

3. In future, we will automate the in-memory optimisation and the recovery from crashes during direct execution.

Q&A