DroidCap: OS Support for Capability-based Permissions in Android

Abdallah Dawoud and Sven Bugiel
Android App Components

- Service
- Activity
- Content Provider
- Broadcast Receiver

Application (UID=A)
Problem: App UID as Ambient Authority

Another app, the Internet, etc.

Application (UID=A)

Service (vulnerable code)

Activity (ad library)

Content Provider

Broadcast Receiver

Device Location

Internet

Icons made by Smashicons, mavadee, Chanut, Dave Gandy from www.flaticon.com is licensed by CC 3.0 BY
App Compartmentalization & Privilege Separation

- Inlined reference monitor
  - But: protected by weak security boundary
App Compartmentalization & Privilege Separation

- Inlined reference monitor
  - But: protected by weak security boundary
- Separate app with distinct UID
  - But: multiple apps installed
- (...)

Diagram:
- Application (UID=B)
  - Activity (ad library)
  - Service (vulnerable code)
- Application (UID=A)
  - Content Provider
  - Broadcast Receiver
  - Location
  - Internet
Our idea:
Represent permissions as object capabilities
Object Capabilities

- **Object Reference**
  - Per-Process
  - Communicable

- **Access Rights**
  - Unforgeable
  - Tamper-proof

**Example file access:** File Descriptor (read only)
How to represent Android permissions as object capabilities?
Calling Services and Permission Enforcement

Application process needs a reference to the SystemService process to call via Binder IPC!
Binder Services: Registration, Discovery, and Invocation

Binder Driver

service_manager

system_server

binder object

Service

userspace

register("location", h1)

register("location", obj)

kernel

binder_ref

h1

binder_node

binder_proc(service_manager)

binder_proc(system_server)
Binder Services: Registration, Discovery, and Invocation

Client
proxy object
h1”

service_manager
<h1, “location”>

system_server
binder object
Service

Binder Driver
binder_proc(client)
binder_proc(service_manager)
binder_proc(system_server)

userspace
find(“location”) find(“location”) dup(h1)

kernel
h1” binder_node
h1” binder_node
h1” binder_node

Service
*binder_object
...
Binder Services: Registration, **Discovery**, and Invocation

Binder Driver

- `service_manager`
- `system_server`
- `client`

Binder Object

- `binder_proc(service_manager)`
- `binder_proc(system_server)`
- `binder_proc(client)`

Proxy Object

- `h1`

BINDER NODE

```
binder_node
```

Access Rights

- Unforgeable
- Tamper-proof

Object Reference

- Per-Process
- Communicable
Binder Services: Registration, Discovery, and **Invocation**

- **Client**
  - proxy object
  - `h1"
  - `Target=h1"
  - `method=…`
  - `params=…`

- **system_server**
  - binder object
  - Service

- **Binder Driver**
  - `binder_ref
  - `h1"
  - `binder_node

- `binder_proc(client)

- `proxy.transact(*data)
Binder Services: Registration, Discovery, and Invocation

Client
proxy object

system_server
binder object

getService

getCallingUID()

Target=h1"
method=…
params=…
callingUid=…

binder_ref
h1"
binder_node

binder_proc(client)

binder_node

*binder_object

…

binder_proc(system_server)

proxy.transact(*data)

stub.onTransact(*data)
Binder Services: Registration, Discovery, and **Invocation**

**Object Reference**
- Per-Process
- Communicable

**Access Rights**
- Unforgeable
- Tamper-proof
Binder Capability

- **Binder capability**: combination of Binder reference and capability fields
- Access rights define the permissions of the capability holder towards the referenced Binder object in binder_node
- Flags and attributes to govern re-delegation and revocation of Binder capabilities
DroidCap: Discovery and Invocation

Client

proxy object

h1"

Service Manager

permissions

<h1, "location">

dup(h1, access_rights=R)

Binder Driver

system_server

binder object

Service

kernel

h1"

accessRights=R

binder_node

client

proxy object

h1"

accessRights=<?>

binder_node

service_manager

binder object

service

userspace

binder_node

binder_ref

h1"

accessRights=R

binder_node

binder_proc(client)

binder_ref

h1"

accessRights=<?>

binder_node

binder_proc(service_manager)

binder_ref

system_server

binder_object

system

binder_proc(system_server)
DroidCap: Discovery and Invocation

Client
proxy object

system_server
binder object

getCallingRights()

Target=h1" method=... payload=... access_rights=R

Client
proxy object

system_server
binder object

getCallingRights()

Target=h1" method=... payload=... access_rights=R

Binder Driver

userspace

kernel

proxy.transact(*payload)

stub.onTransact(*payload)
DroidCap: Discovery and Invocation

Binder Driver

system_server

Client

binder_proc(system_server)

binder_proc(client)

*binder_object

userspace

kernel

DroidCap: Discovery and Invocation

proxy object

proxy.object

h1

binder_node

binder_ref

h1

access_rights=R

binder_node

proxy.object

h1

method=...

payload=...

Target=h1

access_rights=R

- Per-Process
- Communicable
- Unforgeable
- Tamper-proof

Binder Capability ≈ Object Capability
How efficient are Binder capabilities?
Performance

- Android 9, 8, 7.1, and 7.2; Kernel 3.4, 3.9, and 4.1
- HiKey960 device: octa-core 1.8 GHz Cortex-A53 CPU and 3 GB RAM
- Microbenchmarks:

<table>
<thead>
<tr>
<th></th>
<th>Stock Android</th>
<th>DroidCap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binder transaction</td>
<td>34,679 cycles</td>
<td>36,231 cycles (3.41% weighted overhead)</td>
</tr>
<tr>
<td>Permission Check</td>
<td>226.40µs (via IPC)</td>
<td>10.99µs (x7–20 faster)</td>
</tr>
<tr>
<td></td>
<td>77.02µs (local)</td>
<td></td>
</tr>
</tbody>
</table>
Compartmentalizing an App

- Retrofitted open-source Kontalk app to use Binder Capabilities
  - 37 components and 30+ third-party libraries
  - 24 permissions (11 dangerous permissions + Internet)

- Results:
  - 17 Components need no permission
  - 20 components need 1-8 permissions, each.
  - Restricted third-party libraries
    (e.g., TrueTime has no permissions, BarcodeScanner limited to Camera and Internet)
Summary

Our idea:
Represent permissions as object capabilities

Binder Capability

- **Binder capability**: combination of Binder reference and capability fields
- Access rights define the permissions of the capability holder towards the referenced Binder object

DroidCap: Discovery and Invocation

- **Object Reference**
- **Access Rights**
  - Per-Process
  - Communicable
  - Unforgeable
  - Tamper-proof

Binder Capability = Object Capability

Evaluation: Performance

- Android 9, 8, 7.1, and 7.2 - Kernel 3.4, 3.9, and 4.1
- HiKey960 device: octa-core 1.8 GHz Cortex-A53 CPU and 3 GB RAM
- Microbenchmarks:

<table>
<thead>
<tr>
<th></th>
<th>Stock Android</th>
<th>DroidCap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binder transaction</td>
<td>34,679 cycles</td>
<td>36,231</td>
</tr>
<tr>
<td></td>
<td>(3.41% weighted overhead)</td>
<td></td>
</tr>
<tr>
<td>Permission Check</td>
<td>226.40µs (via IPC)</td>
<td>77.02µs (local)</td>
</tr>
<tr>
<td></td>
<td>10.99µs (x7-20 faster)</td>
<td></td>
</tr>
</tbody>
</table>

Thank you! Questions?