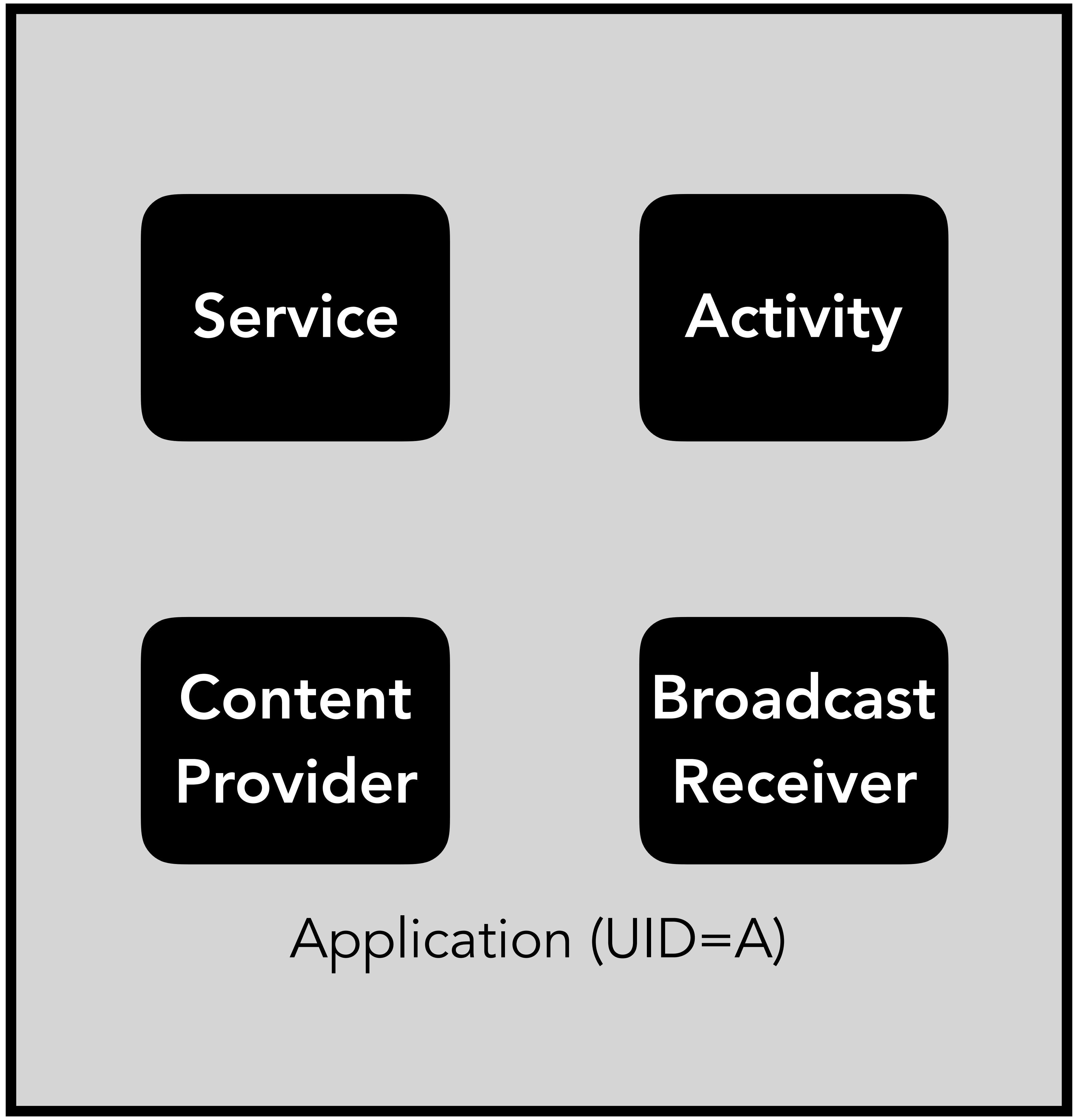
DroidCap: OS Support for Capability-based Permissions in Android

Abdallah Dawoud and Sven Bugiel

Android App Components



Service

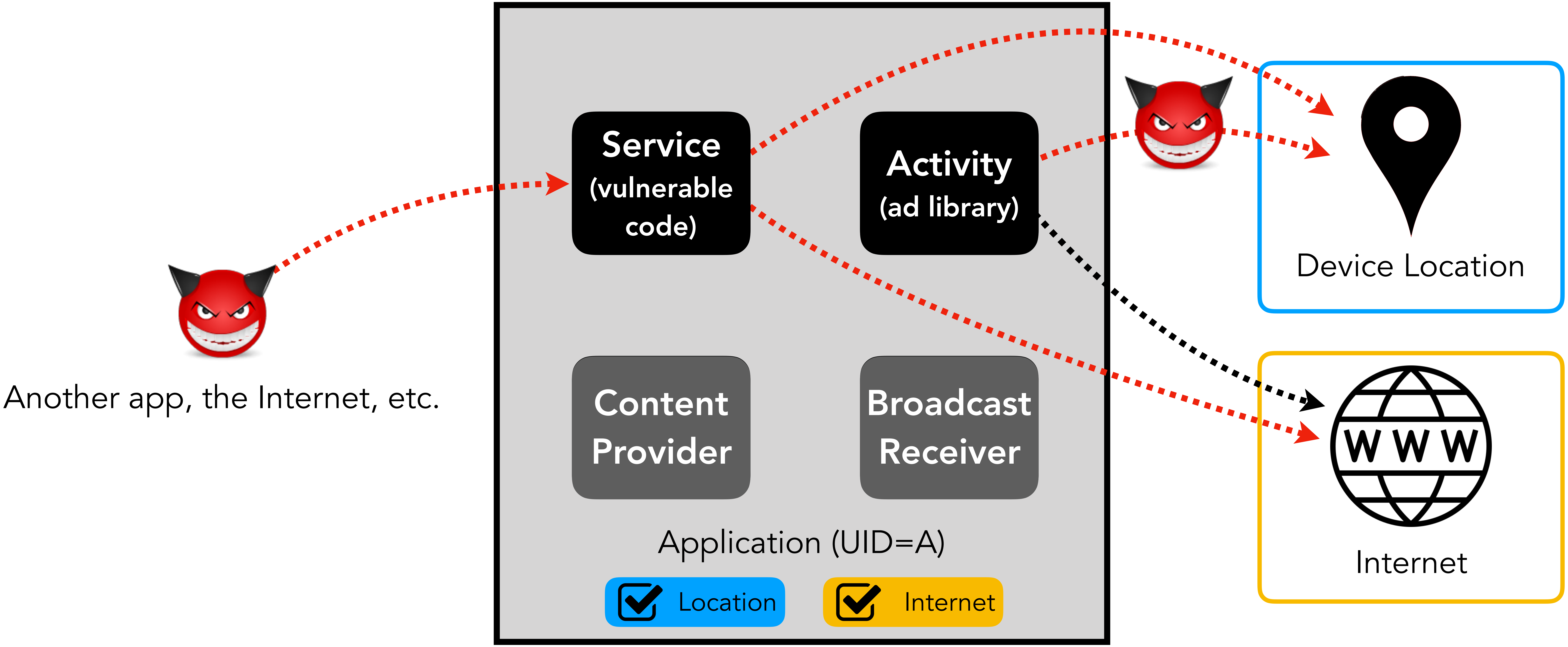
Activity

Content Provider

Broadcast Receiver

Application (UID=A)

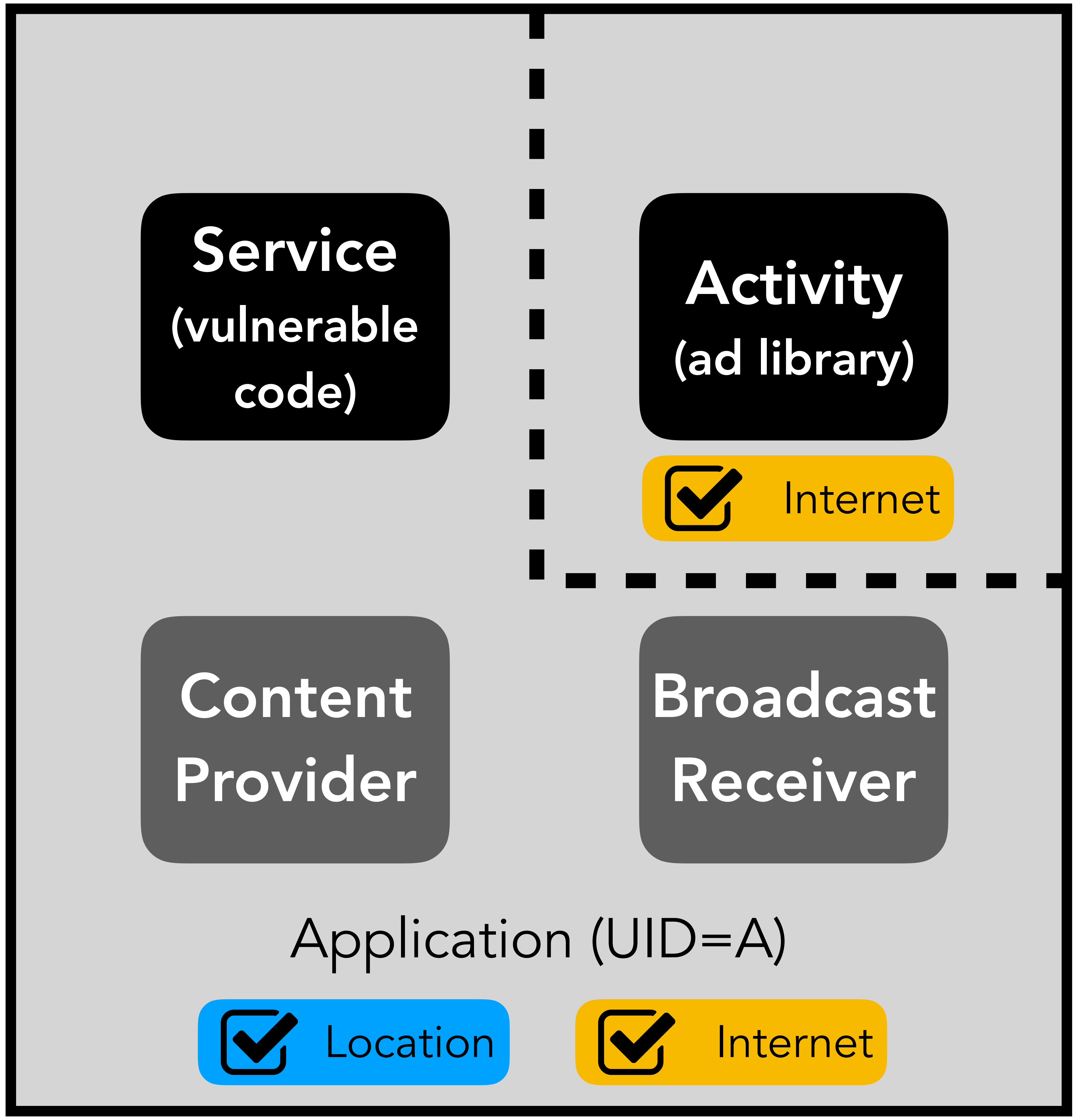
Problem: App UID as Ambient Authority



Icons made by Smashicons, mavadee, Chanut, Dave Gandy from www.flaticon.com is licensed by CC 3.0 BY

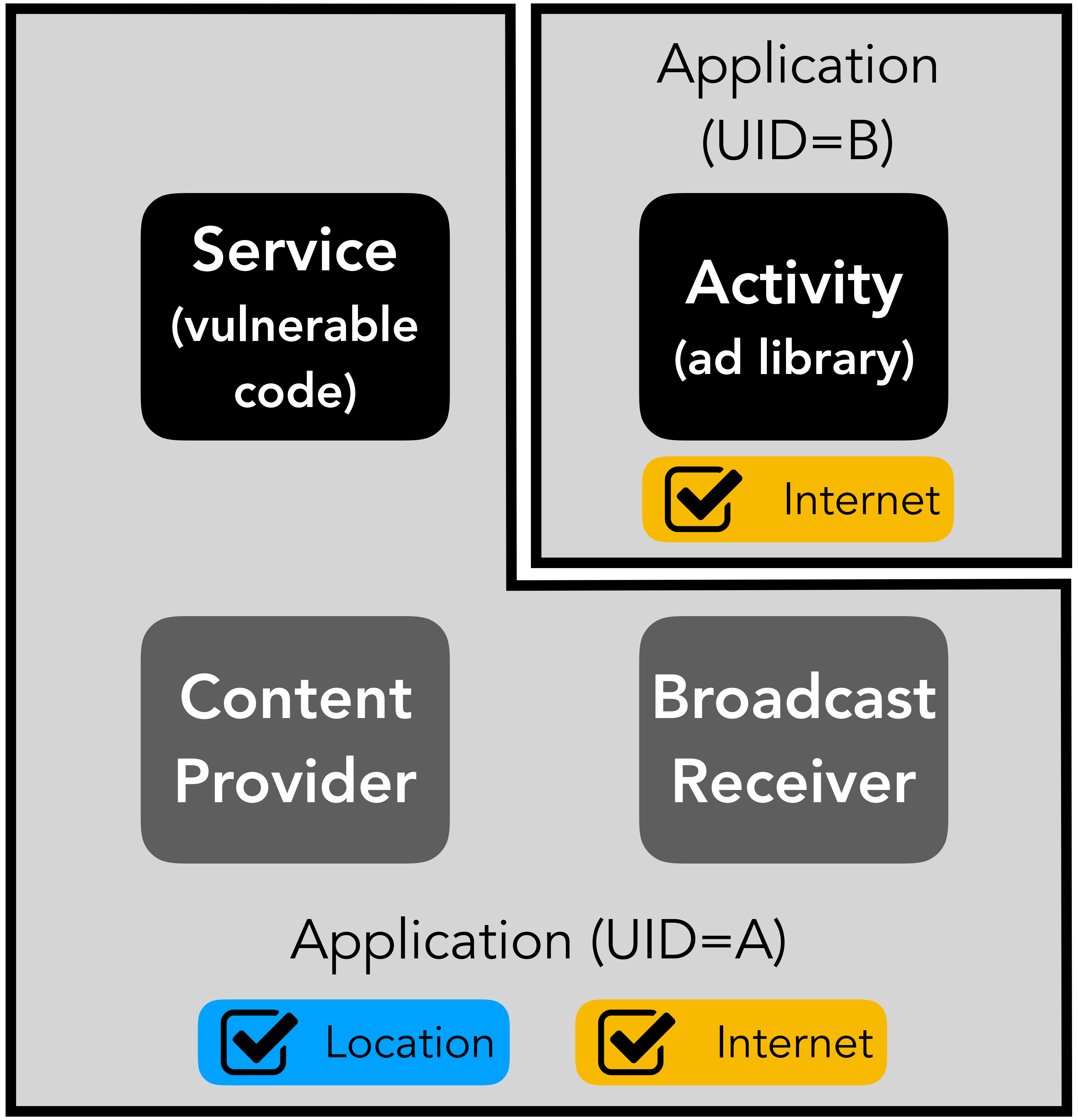
App Compartmentalization & Privilege Separation

- Inlined reference monitor
  - But: protected by weak security boundary



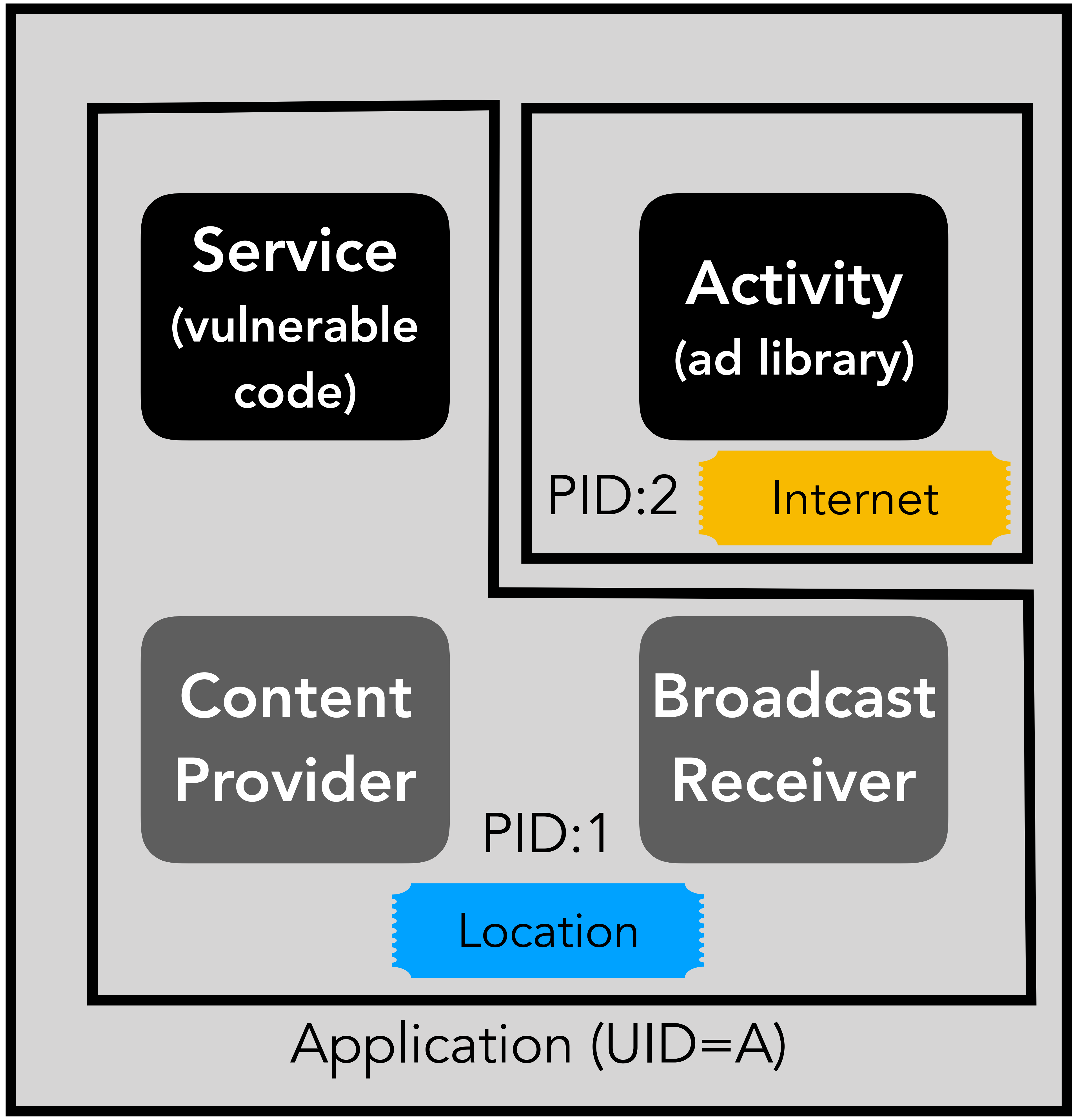
App Compartmentalization & Privilege Separation

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



Our idea:

Represent permissions as object capabilities



Object Capabilities

Object Reference	Access Rights

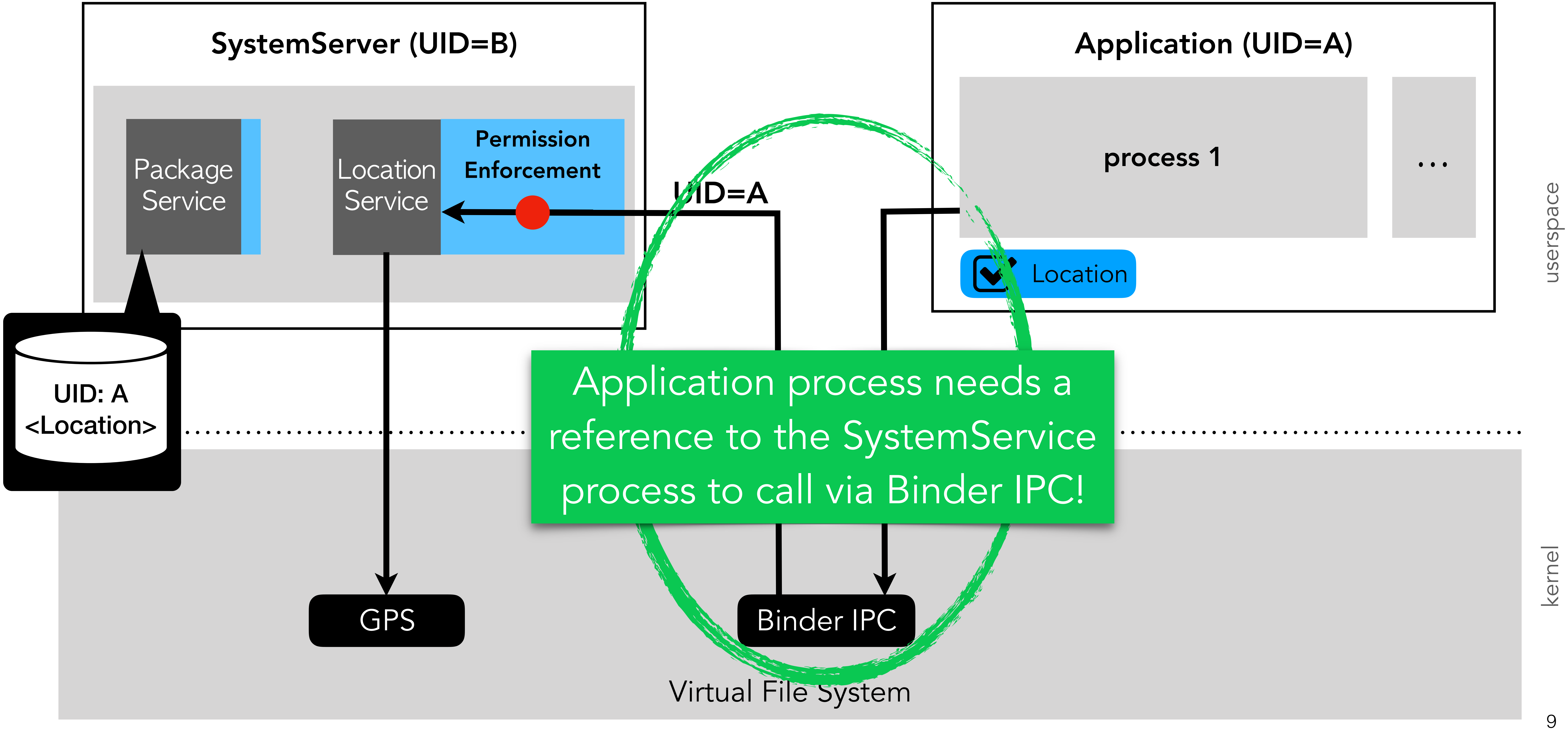
- Per-Process
- Communicable

- 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

userspace
kernel
service_manager
<h1, "location">
system_server
binder object
Service
register("location", h1)
register("location", obj)
binder_ref
h1
binder_node
binder_proc(service_manager)
binder_node
*binder_object
...
binder_proc(system_server)
Binder Driver

Binder Services: Registration, Discovery, and Invocation

userspace
kernel
Client
proxy object
h1”
service_manager
<h1, “location”>
system_server
binder object
Service
find(“location”)
find(“location”)
dup(h1)
h1”
binder_node
binder_proc(client)
binder_ref
h1
binder_node
binder_proc(service_manager)
binder_node
*binder_object
...
binder_proc(system_server)
Binder Driver

Binder Services: Registration, Discovery, and Invocation

Client
service_manager
system_server
proxy obje
h1"
bject
e
userspace
find("location")
fi
kernel
h1"
binder_node
binder_node
bject
...
binder_proc(client)
binder_proc(service_manager)
binder_proc(system_server)
Binder Driver

✓ Object Reference	Access Rights

✓ Per-Process

✓ Communicable

✓ Unforgeable

✓ Tamper-proof

Binder Services: Registration, Discovery, and Invocation

userspace
Client
proxy object
h1"
Target=h1"
method=…
params=…
proxy.transact(*data)
system_server
binder object
Service
kernel
binder_ref
h1"
binder_node
binder_proc(client)
binder_node
*binder_object
…
binder_proc(system_server)
Binder Driver

Binder Services: Registration, Discovery, and Invocation

userspace
kernel
Client
proxy object
h1″
proxy.transact(*data)
getCallingUID()
system_server
binder object
Target=h1″
method=…
params=…
callingUid=…
Service
stub.onTransact(*data)
binder_ref
h1″
binder_node
binder_proc(**client**)
binder_node
*binder_object
…
binder_proc(**system_server**)
Binder Driver

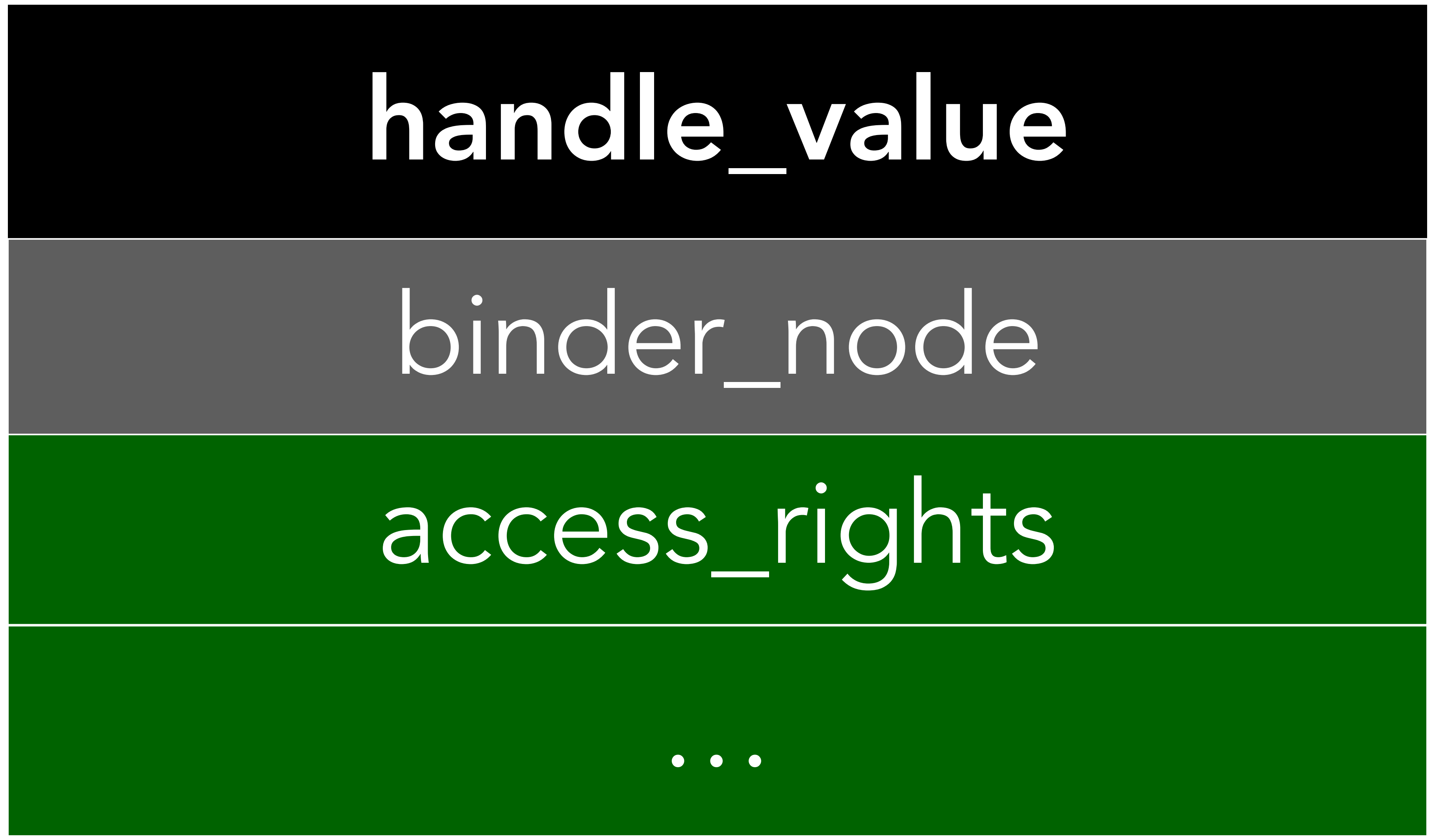
Binder Services: Registration, Discovery, and Invocation

Client
proxy obje
h1″
prox
userspace
kernel
binder_ref
h1″
binder_node
binder_proc(client)
getCallingUID()
system_server
bject
e
bject
...
binder_proc(system_server)
Binder Driver

✓ Object Reference	✗ Access Rights
✓ Per-Process	✓ Unforgeable
✓ Communicable	✓ 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



handle_value

binder_node

access_rights

…

DroidCap: Discovery and Invocation

userspace
Client
proxy object
h1″
service_manager
Permissions
<h1, “location”>
system_server
binder object
Service
dup(h1, access_rights=R)
kernel
binder_ref
h1″
access_rights=R
…
binder_node
binder_proc(client)
binder_ref
h1
access_rights=<?>
…
binder_node
binder_proc(service_manager)
*binder_object
…
binder_proc(system_server)
Binder Driver

DroidCap: Discovery and Invocation

userspace

Client

proxy object

h1"

proxy.transact(*payload)

getCallingRights()

system_server

binder object

Service

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

stub.onTransact(*payload)

kernel

binder_ref

h1"

access_rights=R
...

binder_node

binder_proc(**client**)

*binder_object

...

binder_proc(**system_server**)

Binder Driver

DroidCap: Discovery and Invocation

userspace
kernel
Client
proxy obje
h1″
prox
getCallingRights()
system_server
bject
e
binder_ref
h1″
access_righ
…
binder_no
binder_proc(client)
bject
binder_proc(system_server)
Binder Driver

✓ **Object Reference**	✓ **Access Rights**

✓ 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:

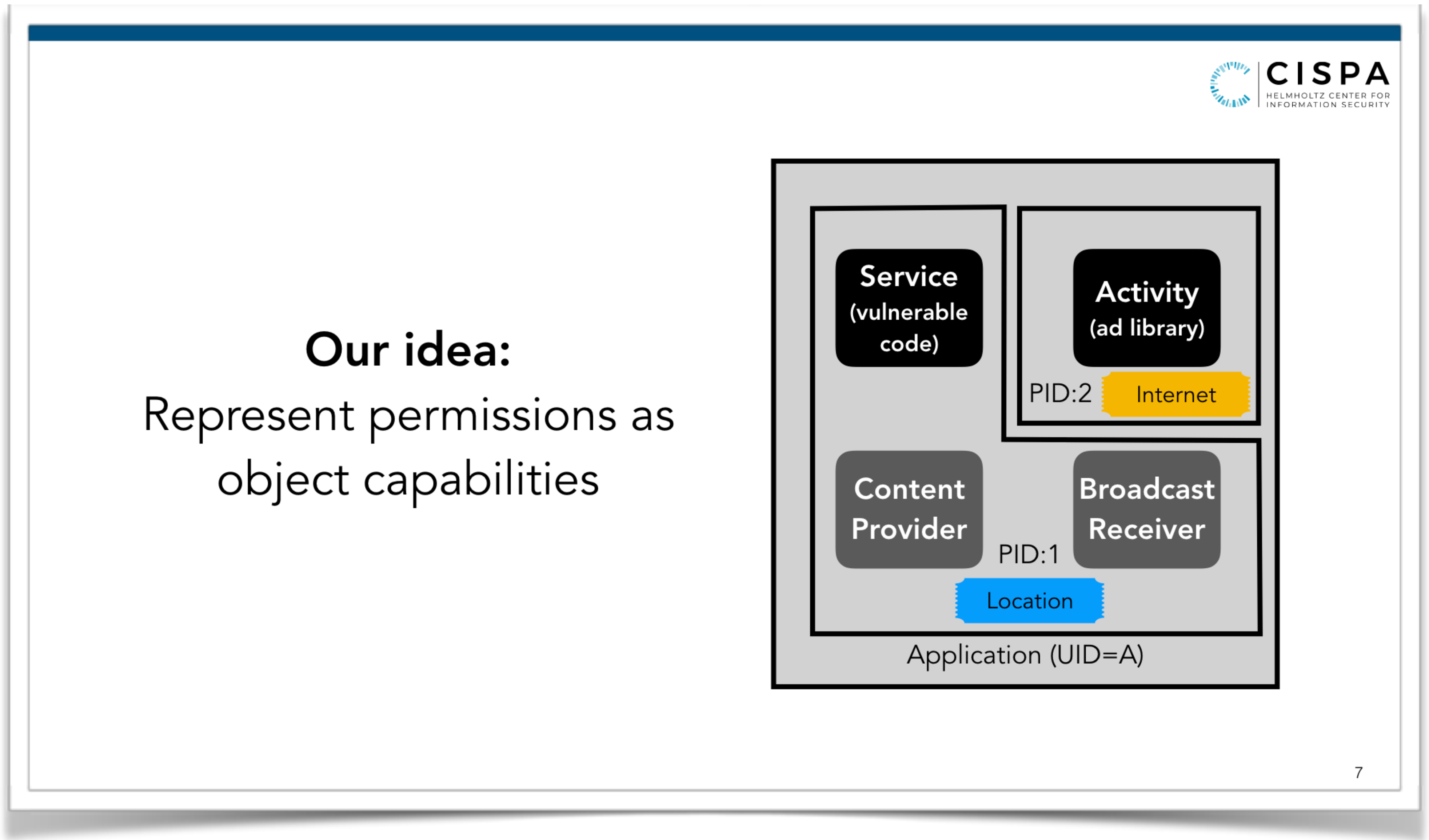
	Stock Android	DroidCap
Binder transaction	34,679 cycles	36,231 cycles (3.41% weighted overhead)
Permission Check	226.40µs (via IPC) 77.02µs (local)	10.99µs (x7–20 faster)

Compartmentalizing an App

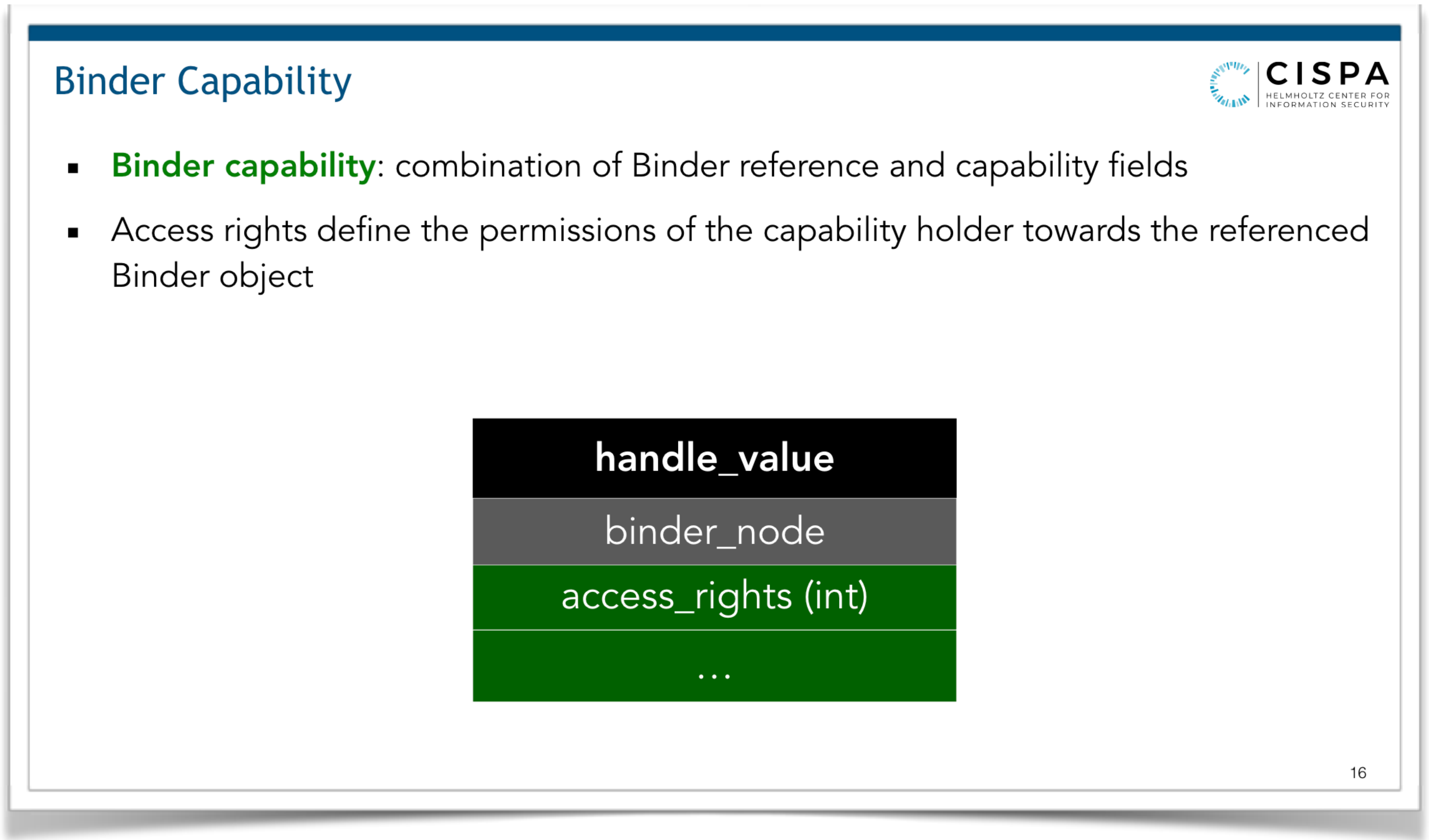
- Retrofitted open-source Kontalk app to use Binder Capabilities
  - 37 components and 30+ third-party libraries
  - 24 permissions (11 dangerous permissions + Internet)

- **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

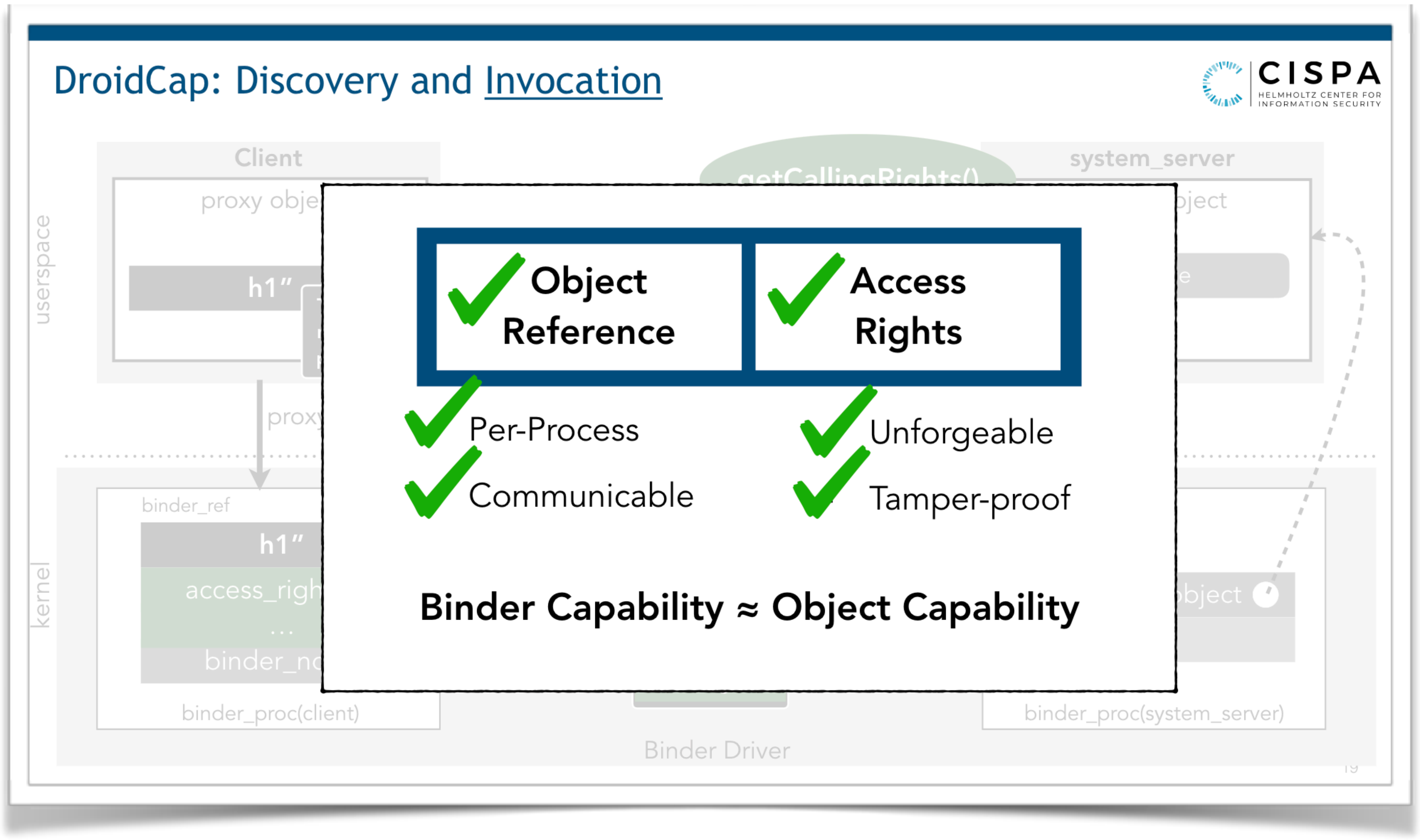


Our idea:
Represent permissions as object capabilities



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

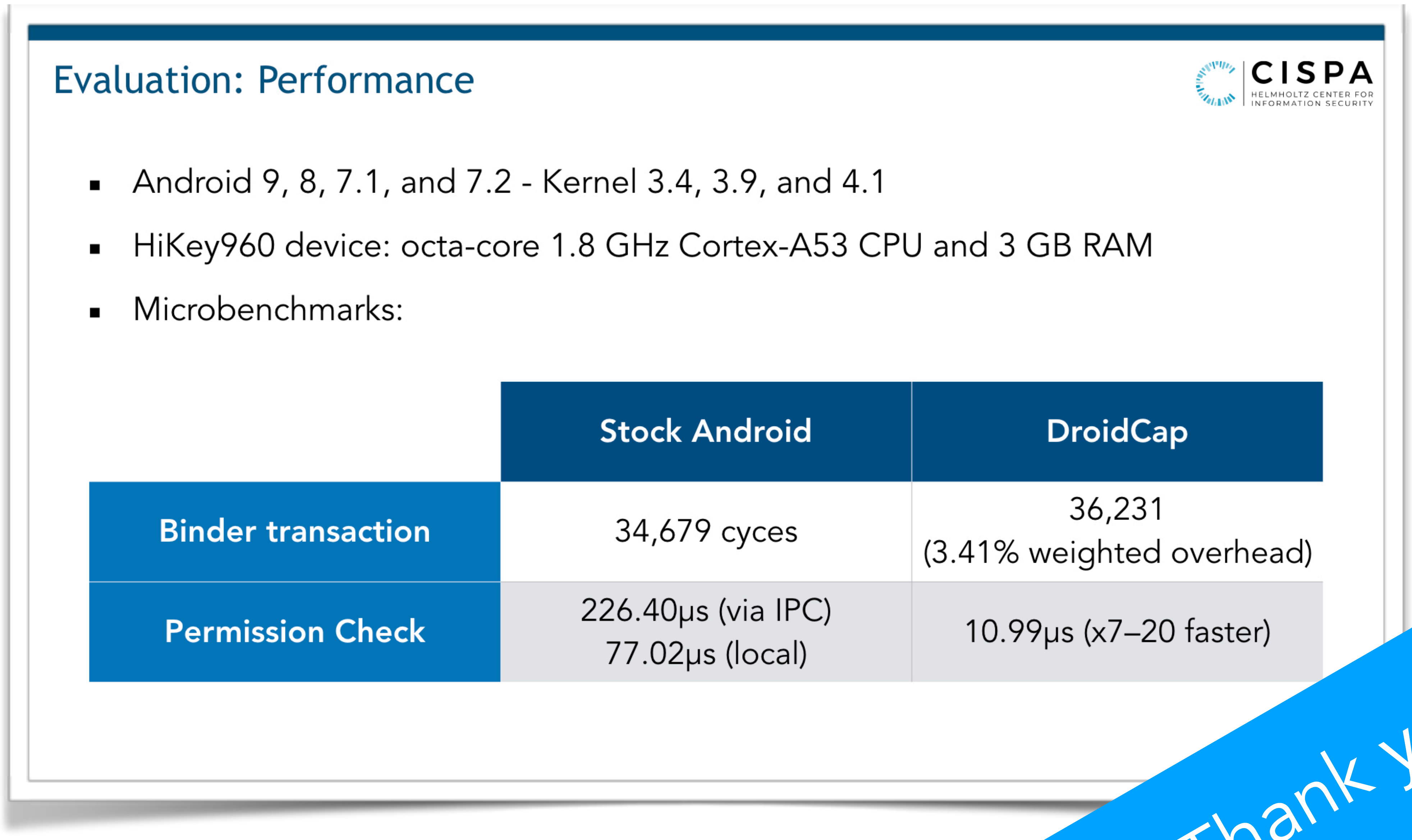


DroidCap: Discovery and Invocation

✓ 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:

	Stock Android	DroidCap
Binder transaction	34,679 cyces	36,231 (3.41% weighted overhead)
Permission Check	226.40µs (via IPC) 77.02µs (local)	10.99µs (x7–20 faster)

Thank you! Questions?