brightsight®

the number one
security lab
in the world

Android Security

Android IPC Mechanism (B2)

Shiqi Li / Brightsight

Security for IPC

In addition to any Linux-type process communication mechanisms such as file system, local sockets, signals, Android provides new IPC mechanisms:

- Linux way
  - Sockets
  - Shared files

- Android way
  - Intents
    - can apply permissions w.r.t. sender and define receivers
  - Binder
    - AIDL

Files as IPC

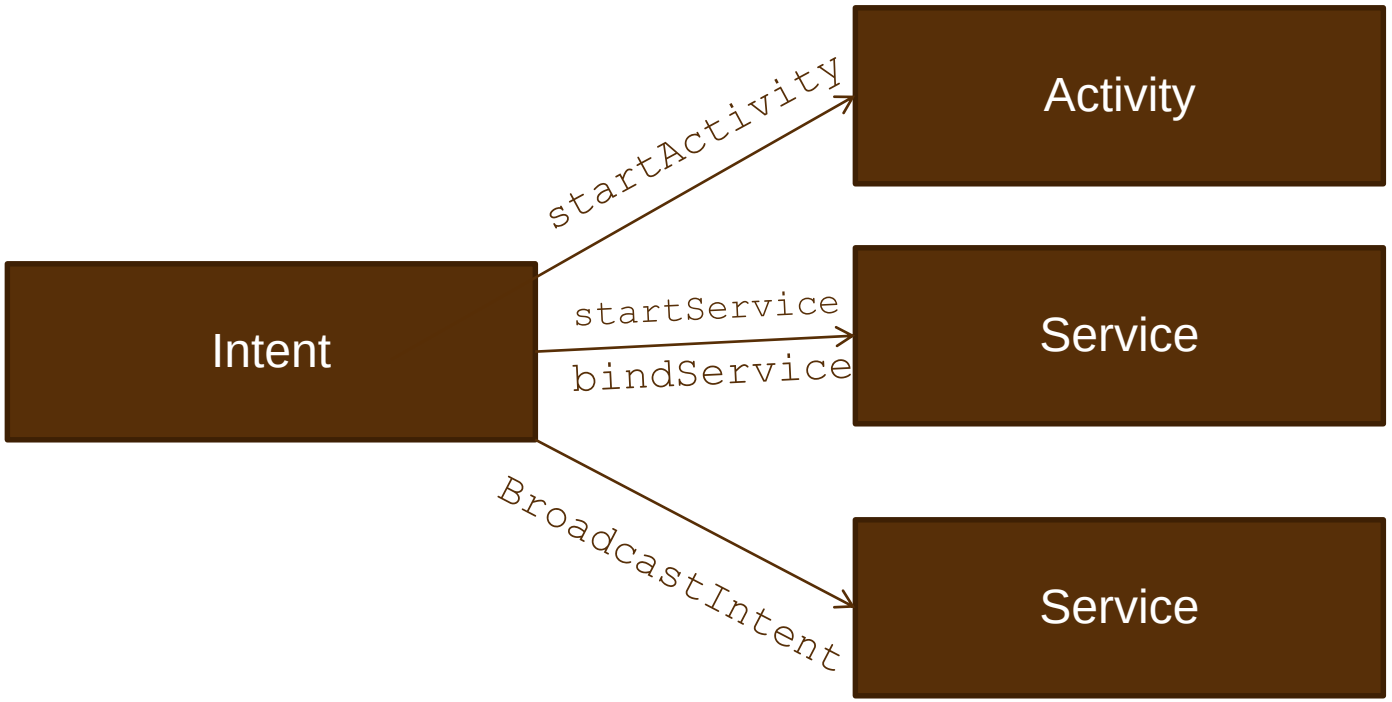
Processes can use shared–preferences Files as a way to implement IPC, This can be achieved in two ways:

- Setting the flag MODE_WORLD_READABLE while creating the file
- Use the `android:sharedUserId` tag in the application manifest

The use of the file-system as IPC is discouraged.

Intents as IPC

Intent messaging is a framework for asynchronous communication among Android components (usually application layer). The context object can make use of Intents to communicate whit different processes, the method `putExtra` can be used to send data to the server process



Intents as IPC (2)

Only the `exported` components can be started by external intents, By default, components can only receive internal Intents
(note: `broadcastReceivers` are **exported by default**)

Intents can be restricted by using application permissions

- Services and Activities
  - Should use <permission> tag to require permissions from the clients
- BroadcastReceivers
  - Intents can limit the broadcast receivers capable of receiving the broadcast message by defining a not-null permission in the method call `broadcastIntent (Intent, Permission)`

Intents – Security Issues

In order to support communication between different processes, Android provides a simple inter-process communication (IPC) mechanism which allows different process to share information.

There are two main security risk involved with the use of intents:

- For Broadcast messages if sender process does not correctly specify recipient, Attacker can intercept the message
- If access permissions or caller authorization is not performed properly. Attacker can hijack activity, service, broadcast via intent to lead to a malicious target.

Binders and Services

- Services can override the `onBind` method in order to provide binder interfaces to the clients
- A service registers itself to service manager by providing its Binder and a name. Any client can obtain its Binder reference from `servicemanager` by query its name.
  - On the client side Binders can be obtained by calling the `context.bindService` method with an Intent.
- Binders are then casted to the proper object by using the Interface declared by the AIDL definition.
- Service Binders can be protected by Android permissions

System (Global) Services

System(Global) services can be registered and obtained by the following methods:

```
ServiceManager.addService
ServiceManager.getService
```

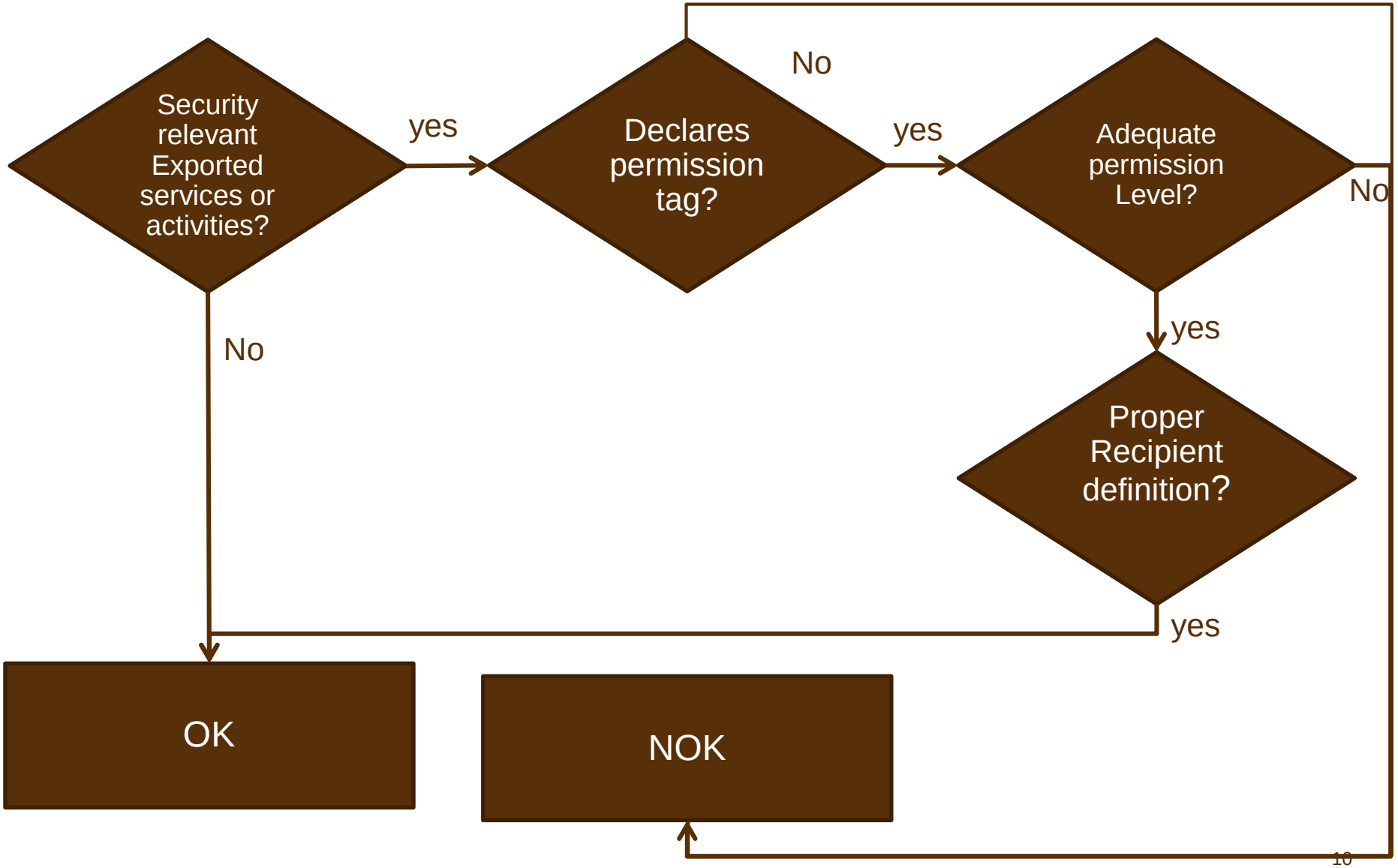
System services are not protected by static permissions but they can implement security by verifying the caller identity and permissions:

- `android.os.Binder.getCallingUid`
- `PackageManager.getPackagesForUid(int uid)`
- `PackageManager.getPackageInfo` (with the `PackageManager.GET_PERMISSIONS` flag)

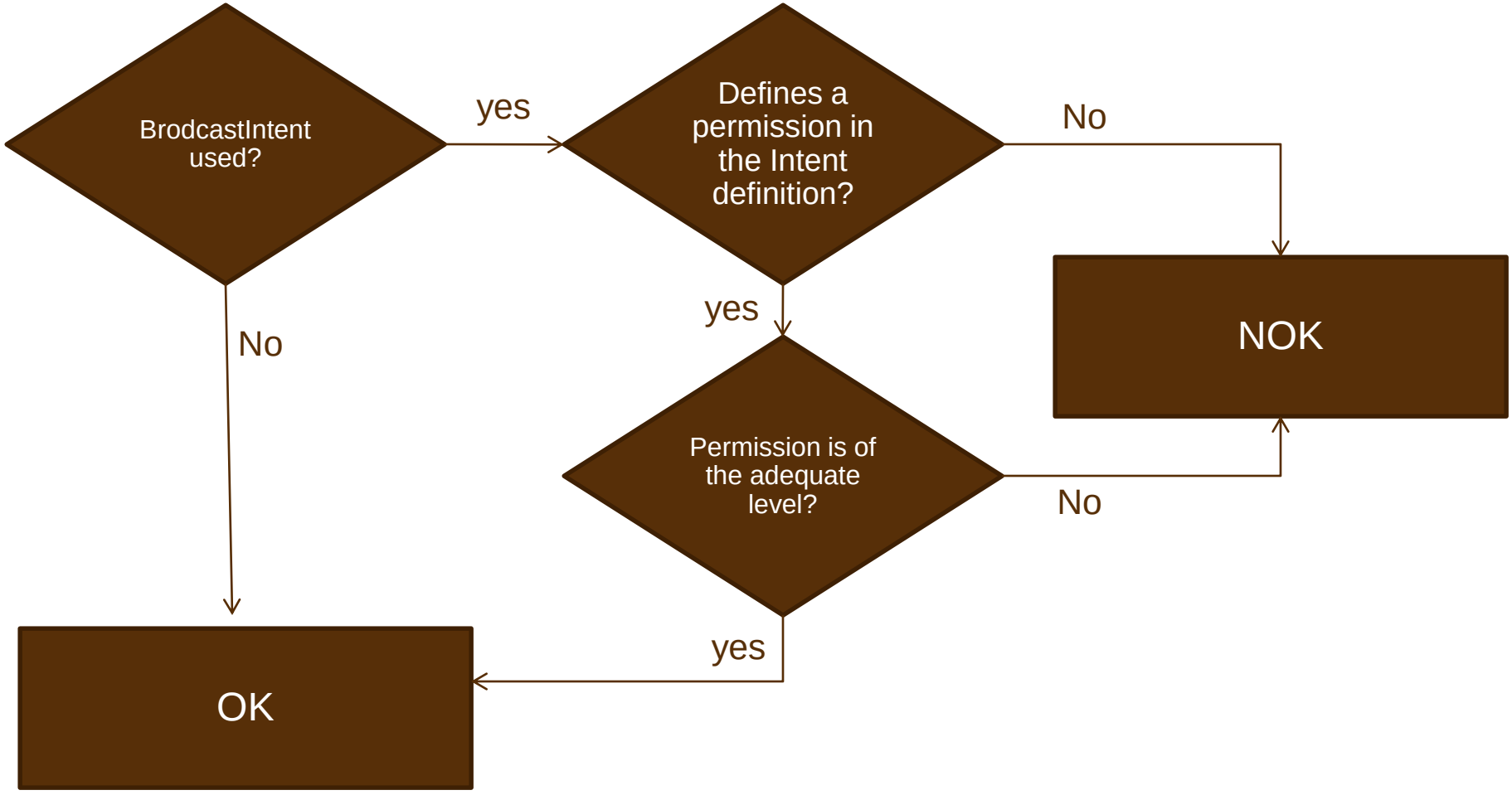
Security Evaluation Checklist

- Is there any world Readable-Writable file?
- Is there any Exported component in the manifest.xml file?
- Is there any .aidl file included in the /src folder?
- For the exported Components is there any permission required <uses-permission> tag? Note: BroadcastReceiver components are exported by default.
- What is the level of the requested permissions for exported components objects?
- (Suggested permission levels are signature or systemandSignature)
- Are the Binder Interfaces implementing any access control check?
- Are the intents specifying the proper recipients by setting both the intent.setComponent and intent.setAction values?

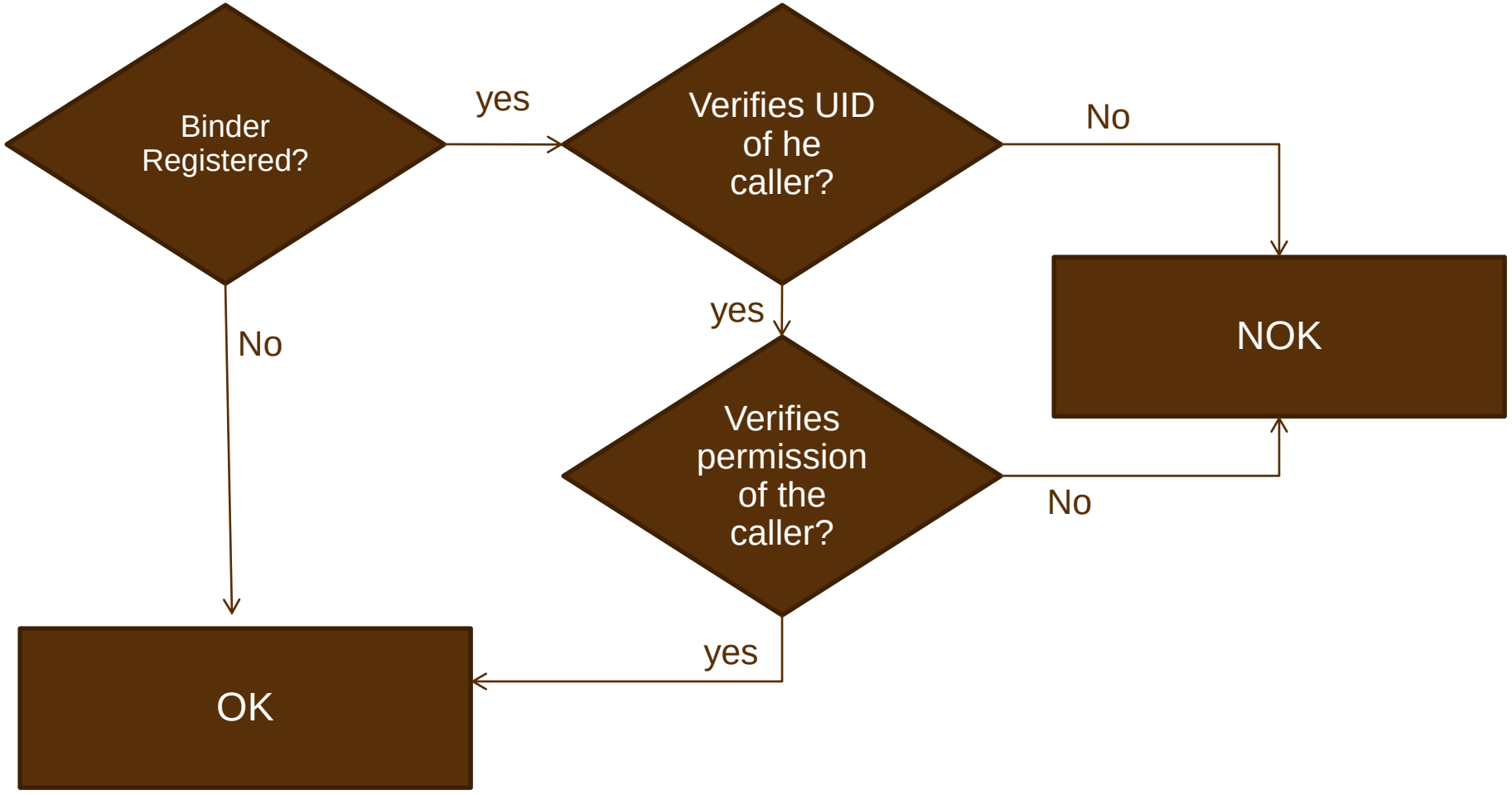
Evaluate Exported Components



Evaluate Broadcast Receivers



Evaluate Global Binder Service



PCI PTS security concerns

DTR B2:

The evidence shall

- enumerate all logical and physical interfaces by the POI
  - Interfaces configuration to accept commands (command interpreter, API, exported Binder, BroadcastReceiver)
  - List all API on all of the logical interfaces
- show that only documented commands implemented.
- Detail method verification length and content of CMD before processing.
- If non-firmware can be executed, provide buffer overflow analysis/testing
- Fuzzing on selected interfaces
- Identify all command interpreters within the SW

DTR A4: secure service protection

- If access to secure CPU is implemented with a System Service, whether proper the application is properly verified.
- If virtual PINPAD is implemented with a System Service + System Application, whether the plaintext PIN is properly protected inside the Secure Service. Also if IPC is used to transfer plaintext PIN (e.g. Virtual PINPAD → /dev/<secure CPU> → PIN encryption in secure CPU; or VPP→CryptoService), whether this link is secured?

Questions?