Mobile Phone Security

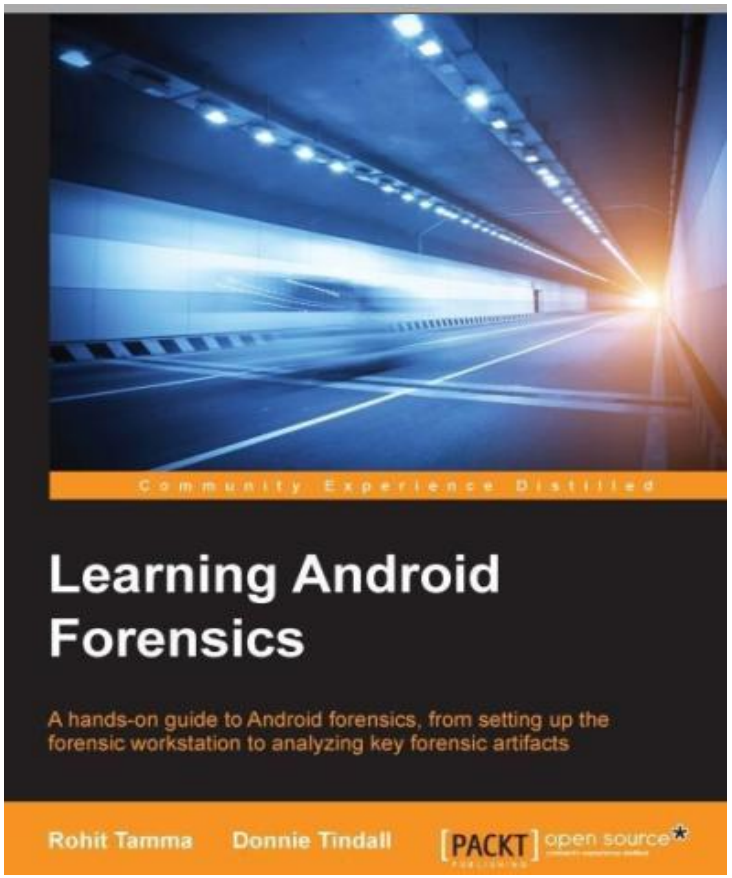
Dr. Digvijaysinh Rathod
Associate Dean
School of Cyber Security and Digital Forensics
National Forensic Sciences University

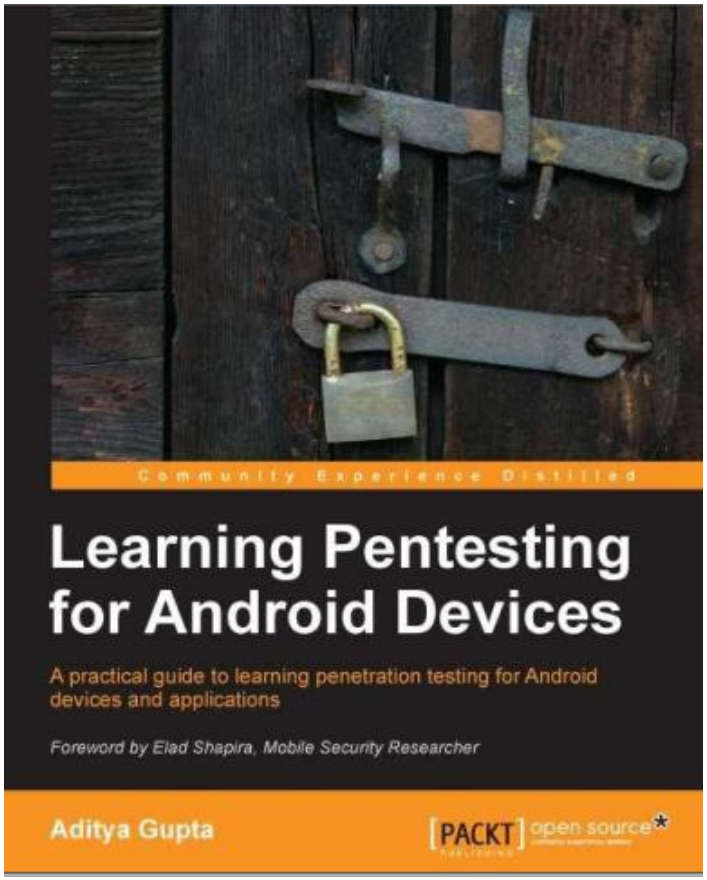
digvijay.rathod@nfsu.ac.in

Android Sanboxing

Secure inter-process communication

Reference





digvijay.rathod@gfsu.edu.in

Application sandboxing

✓In order to isolate applications from each other

✓Android takes advantage of the **Linux user-based** protection model.

✓In Linux systems, each user is assigned a **unique user ID (UID)** and users are segregated so that one user does not access the data of another user.

✓All resources under a particular user are run with the same privileges.

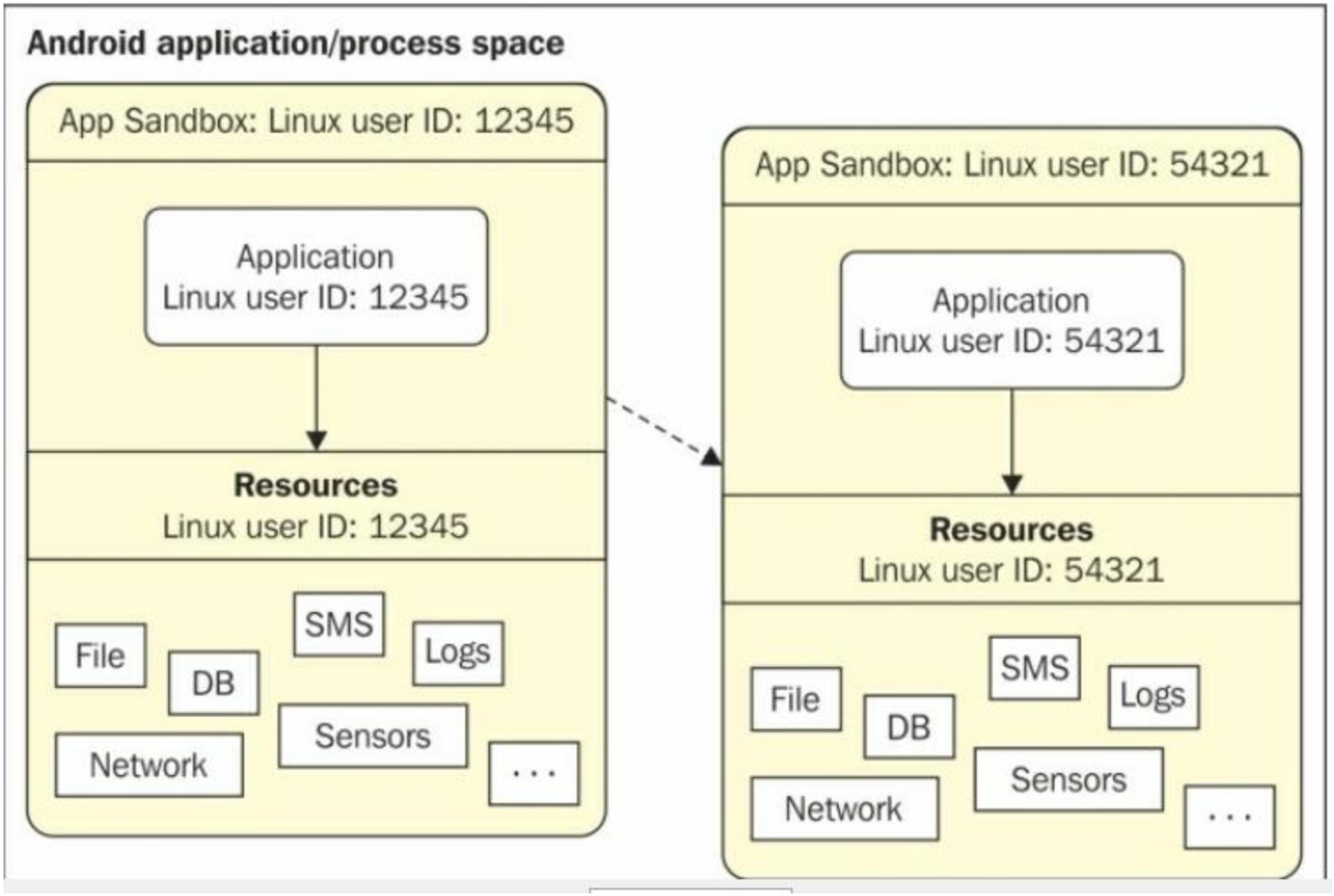
Application sandboxing

✓Android application is assigned a **UID and is run** as a separate process.

✓What this means is that even if an installed application tries to do something **malicious**, it can do it only within its **context and with the permissions it has.**

✓By default, applications cannot **read or access the data** of other applications and have limited access to the operating system.

Application sandboxing



Two applications on different processes on with different UID's

What is Android Broadcast Receiver?

✓Since the application sandbox mechanism is implemented at the **kernel level,** it applies to both **native applications and OS applications.**

Secure inter-process communication

✓As discussed in the above sections, sandboxing of the apps is achieved by running apps in different processes with different Linux identities.

✓System services run in separate processes and have more privileges.

✓Thus, in order to organize data and signals between these processes, an **inter-process communication (IPC) framework is needed**.

✓In Android, this is achieved with the use of the Binder

Secure interprocess communication

✓The **Binder framework** in Android provides the capabilities required to organize **all types of communication between various processes**.

✓Android application components, such as **intents** and **content providers**, are also built on top of this Binder framework.

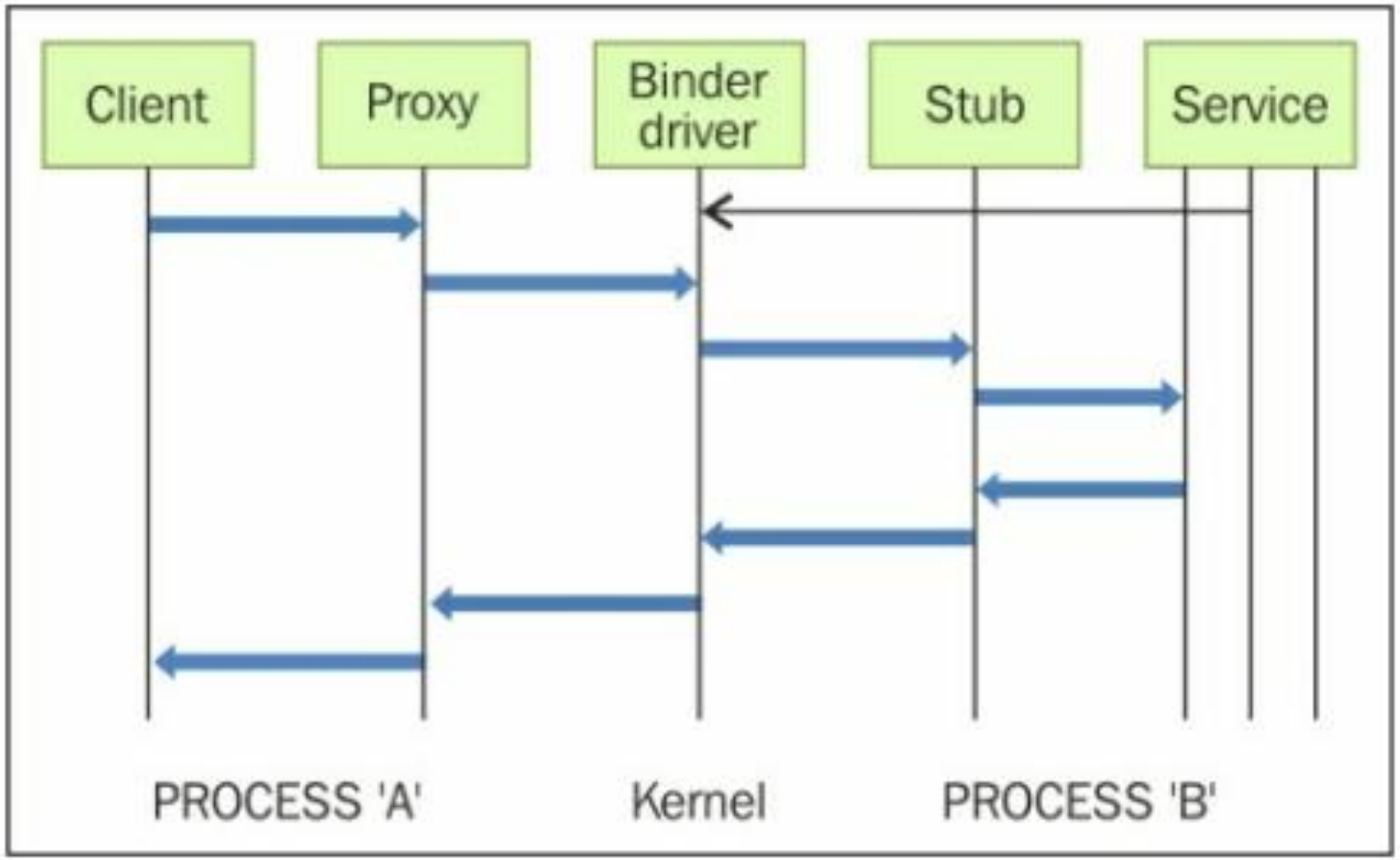
✓Using this framework, it is possible to perform a variety of actions such as **invoking methods on remote objects.**

✓Let us suppose the application in Process 'A' wants to use certain behavior exposed by a service which runs in Process 'B'.

✓In this case, **Process 'A' is the client and Process 'B' is the service.**

✓The communication model using Binder is shown in the following diagram:

Secure interprocess communication



Binder Communication Model

Secure interprocess communication

✓All communication between the processes using the Binder framework occurs through the **/dev/binder Linux kernel driver**.

✓The permissions to this **device driver are set to world readable** and writable.

✓Hence, any application may write to and read from this device driver.

✓All communications between the client and server happen through **proxies** on the **client** side and **stubs** on the **server side**.

1. To get this process working, **each service** must be registered with the **context manager.**
2. the **context manager** acts as a **name service**, providing the handle of a service using the name of this service.
3. Thus, a client needs to know only the **name of a service** to communicate.
4. Each service (also called a Binder service) exposed using the **Binder mechanism** is assigned with a **token**.

1. This **token is a 32-bit** value and is **unique across all processes in the system.**
2. A client can start interacting with the **service** after discovering this value.
3. This is possible with the help of **Binder's context manager.**
4. the context manager acts as a name service, providing the handle of a service using the name of this service

1. The name is resolved by the context manager and the client receives the token that is later used for communicating with the service.

Mobile Phone Security

Dr. Digvijaysinh Rathod
Associate Professor
(Cyber Security and Digital Forensics)
Institute of Forensic Science
Gujarat Forensic Sciences University

digvijay.rathod@gfsu.edu.in