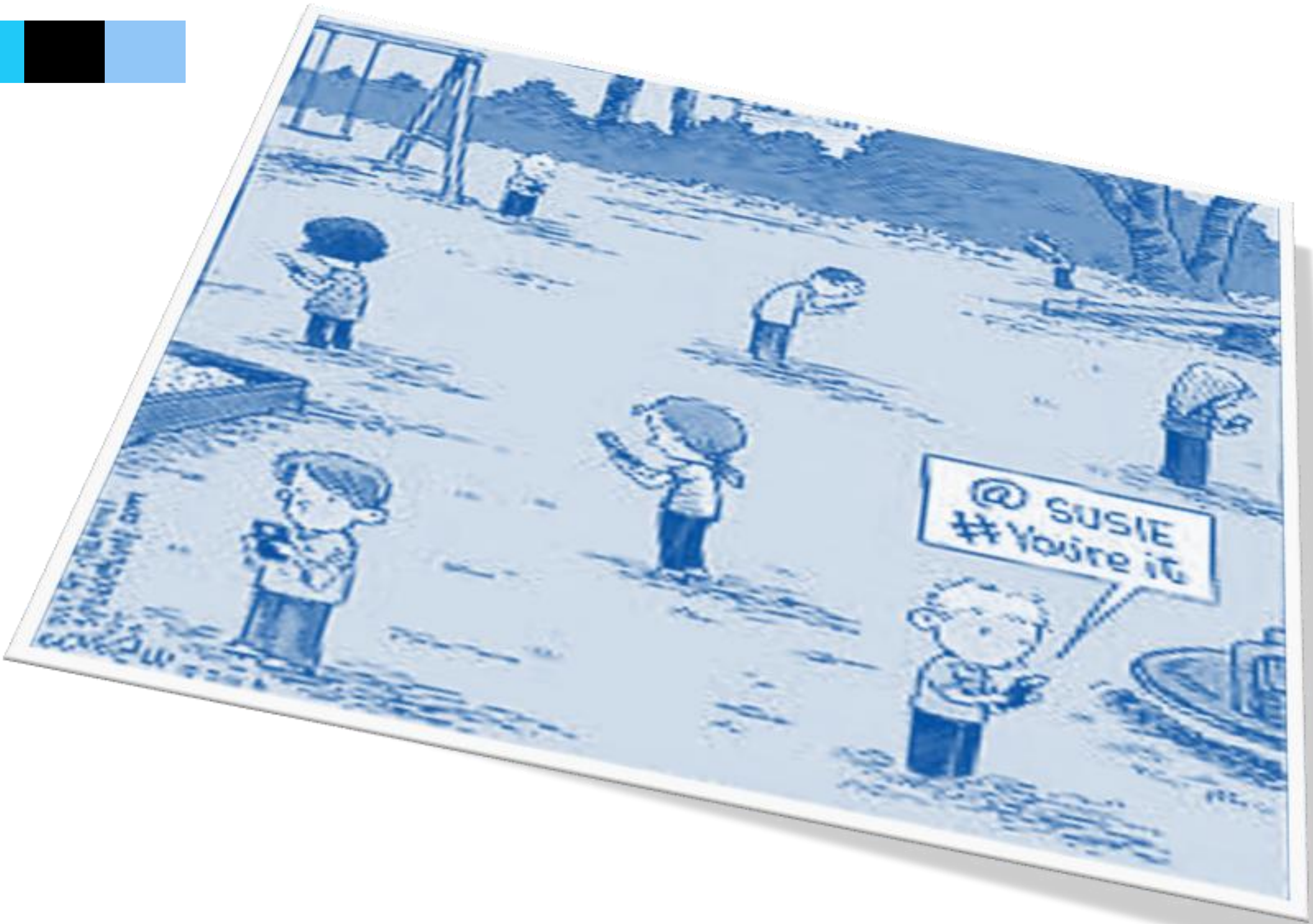
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

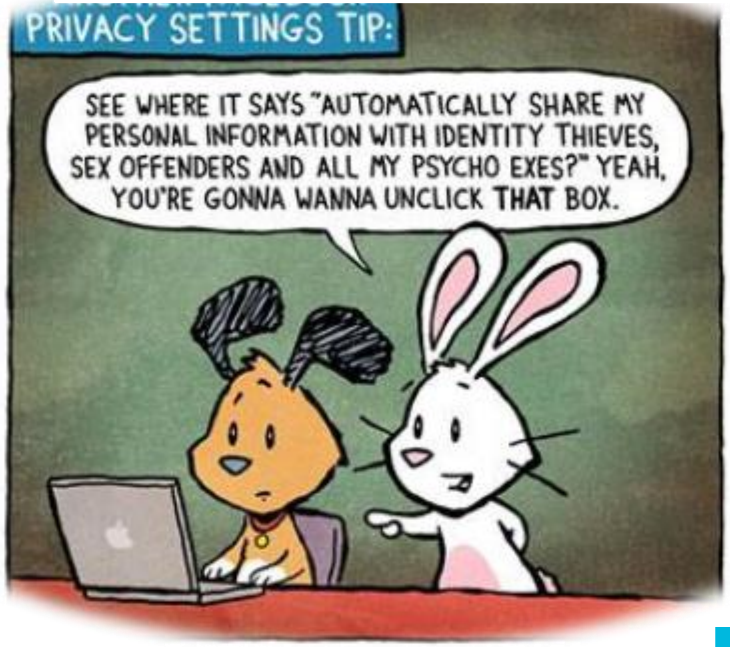
Privacy Violations Detection in Android like Systems

Nora Cuppens
Directrice de Recherche

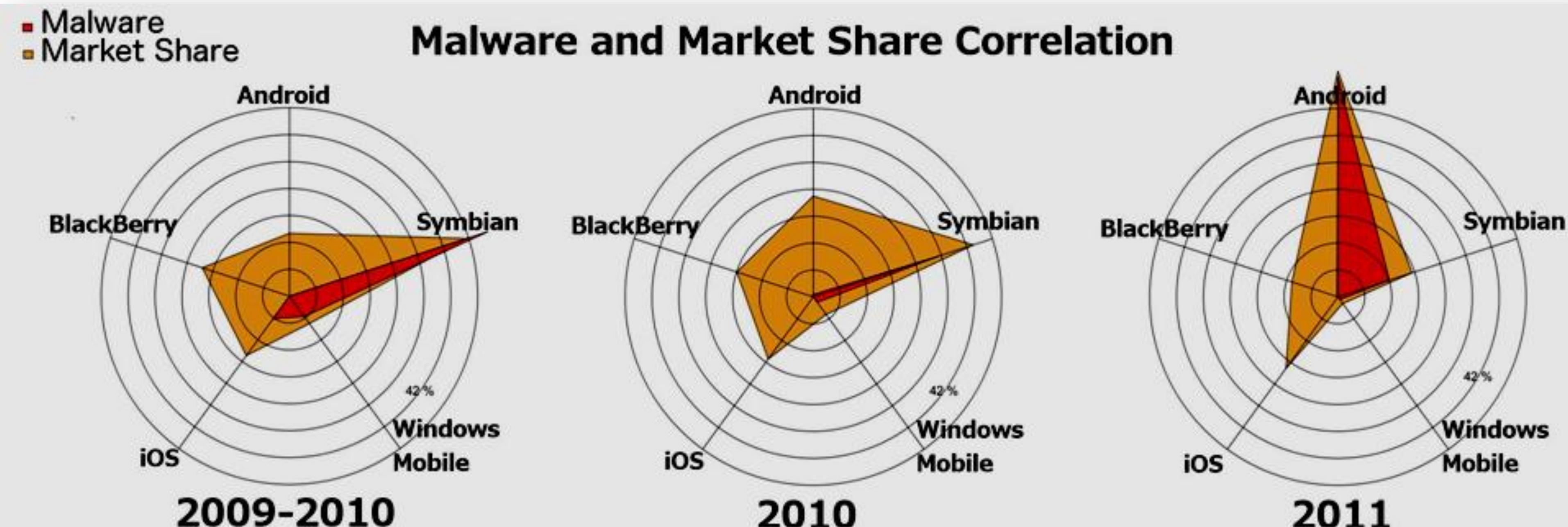
Colloque IMT'2017, November 10th


@ SUSIE
#You're it


SIRI- TELL ME YOUR INNERMOST SECRETS — JUST THIS ONCE.
Sorry, G-man, but if I did that, I wouldn't be a smart phone, would I?
Apple
FEDS


PRIVACY SETTINGS TIP:
SEE WHERE IT SAYS "AUTOMATICALLY SHARE MY PERSONAL INFORMATION WITH IDENTITY THIEVES, SEX OFFENDERS AND ALL MY PSYCHO EXES?" YEAH, YOU'RE GONNA WANNA UNCLICK THAT BOX.

Why targeting Android?



- **2017**
  - 71% - 87% market share
  - 2.7+ billion apps, 70+ billion downloads (Src: Google)
  - 1M+ Android devices activated everyday (Src: Google)
- **Ideal platform for security research**

* *Juan Tapiador*

Informal problem statement: – How invasive Android Apps are?

- **Uber: knows everywhere you go**
  - Tracking customers
- **Whisper, Yik-Ya: supposedly anonymous**
  - “De-anonymizing users and take control of the account …”
- **Angry Birds: only a game?**
  - User profiling
- **Snapchat: self-destructing photo app that doesn't**
  - Hacked and lost a database of several million usernames connected to phone numbers.
- **Brightest Flashlight: flashlight apps**
  - Exploiting their phone's internet connection in order to deliver targeted advertising

Informal problem statement:
– Data leakage / Privacy loss

- **Android type system offers a nominal security solutions**

- **Progress has been made in this area**
  - Access control
  - Data flow control

- **Our Research works**
  - Solving under-tainting problem
  - Detecting flows in JNI
  - Dealing with side channel attacks
  - Detecting / Reacting activity hijacking

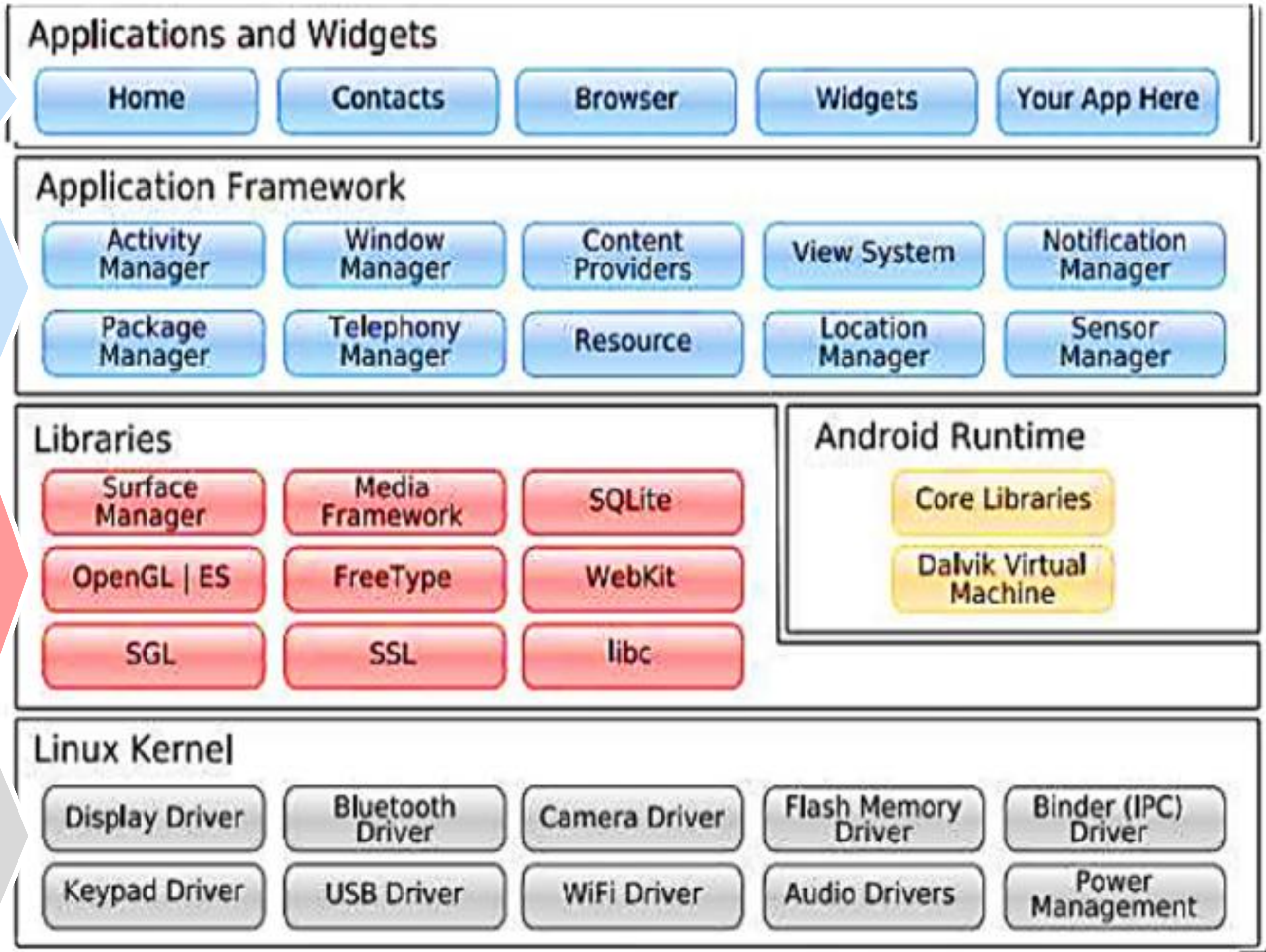
Couple of reminders:
– Android System Architecture and Security

- System Applications
- User Applications

- IPC reference monitor
- Sandboxing
- Permission levels

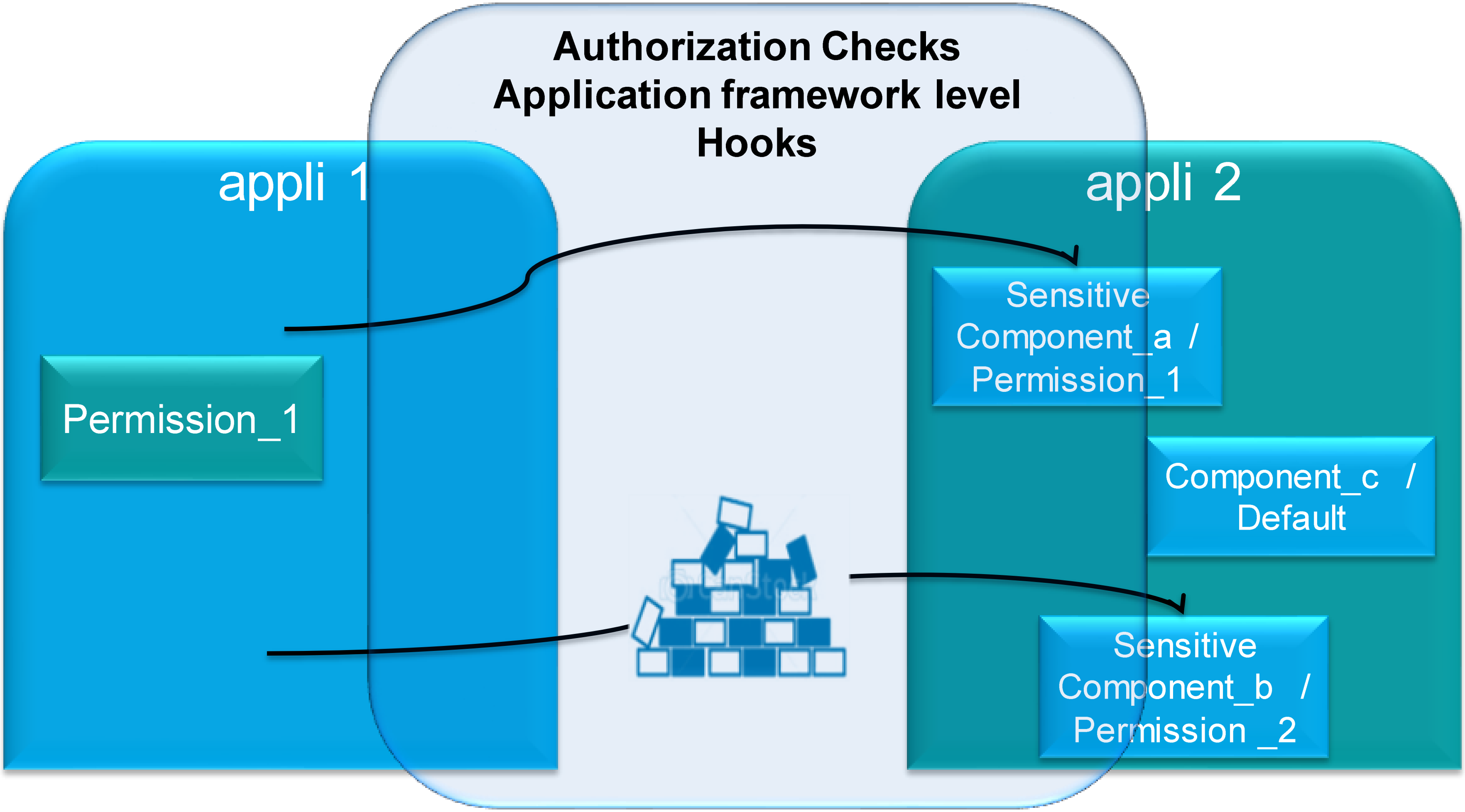
- Secure boot
- Secure file system
- Native executables protection

- Discretionary AC
- Application Sandbox



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Couples of reminders:
– Security in the Application Level



Some Weaknesses of Android Security Model

- Revocation limits
- Few sources for applications, warnings about security implications displayed during run-time
- Flawed permissions model
- Malware obfuscated inside legitimate-looking applications
- Google play store: insufficient control
- Applications isolation: malicious k-ary applications
- Tricky problem of Patching / Updating

Android Security – Enhanced solutions

Access Control

Data Flow Control

Access Control in Android Systems – The progress

- **Applications certification**
  - Kirin (Enck et al.)]
  - Avoid manual certification by code inspection (SymbianSigned, Apple)
  - Provide lightweight certification based on predefined rules at install-time
- **Application access control policy at install**
- **Application inter-communications security policy at execution**
  - [Saint (Ongtang et al.)]
  - Managing authorization assignments and their use at run-time
  - In accordance with the application provider policy
- **Dynamic control of permissions granted to applications**
  - [Apex (Nauman et al.)]
  - The user chooses the permissions to be granted to the applications and imposes constraints on the use of resources

– Access control is not sufficient

- … of course
- Does not address the data flow problem

Information flow

- A command sequence implies an information flow from `x` to `y` if the value of `y` after performing this sequence makes it possible to infer information on the value of `x` before the execution of this sequence

- ```
  boolean b := <secret>
  if (b) {
  x := true; f();
  ```

Information flow from `b` to `x`
  ```

Data flow control: static analysis – Reminder

- **Analyzing the code without executing it**
- **Performed at the install-time or compile-time**
- **Performed on the source or on the bytecode**

Data flow control: static analysis

Static analysis of Dalvik bytecode of applications	Tracking flows between URIs to generate constraints on permissions	Requiring the source code, Packaged applications are not considered	ScanDroid [Fuchs et al.]
Analyzing applications before making them available	Analysis of decompiled DEX files to discover vulnerabilities based on intents exchanges	Secure communication: No formal guaranties	ComDroid [Chin et al.]

Only Explicit flows are considered

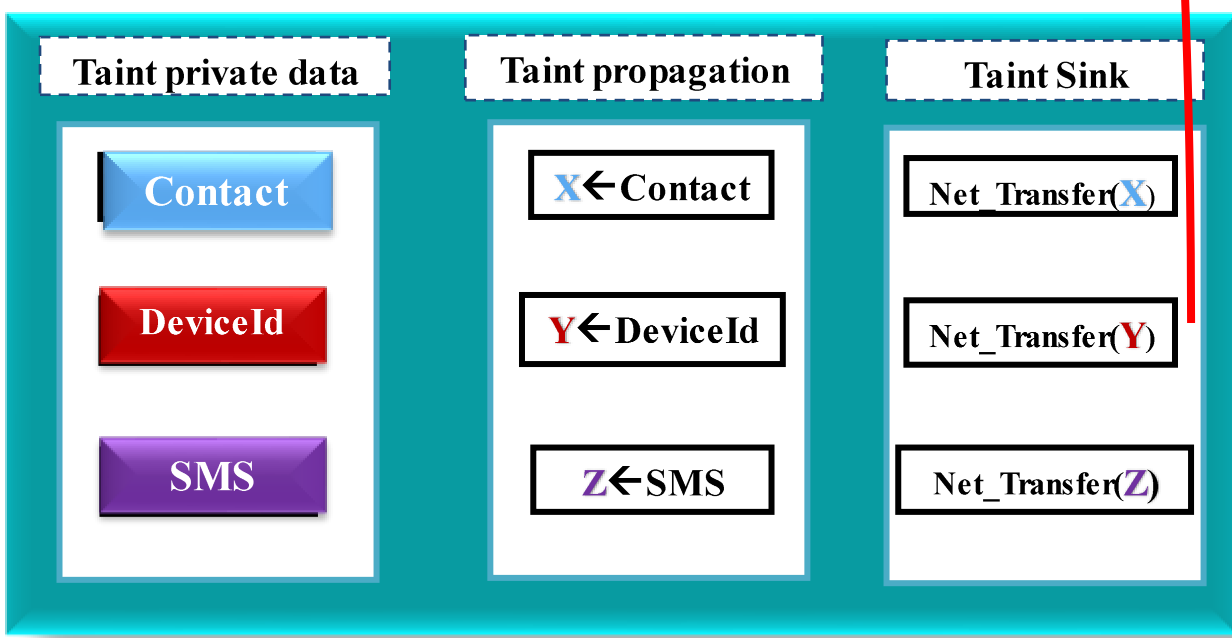
Data flow control: dynamic analysis – Reminder

- Instrumentation of the code before its execution
- Analysis performed at run-time
- Binary code Tracking

Data flow control: – “Tainting” based dynamic analysis

- **TaintDroid [Enck et al.]**
- **Tainting**
  - Technique for tracing dependencies of information from a given point

```
x  = taint ()
…
y = z + x
…
Sent_Net(y)
```



“Tainting” propagation logic – Examples

Op Format	Op Semantics	Taint Propagation	Description
const-op v_A C	$v_A \leftarrow C$	$\tau(v_A) \leftarrow \emptyset$	Clear v_A taint
move-op v_A v_B	$v_A \leftarrow v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
move-op-R v_A	$v_A \leftarrow R$	$\tau(v_A) \leftarrow \tau(R)$	Set v_A taint to return taint
re[illegible]	[illegible]	[illegible]	[illegible]
m[illegible]	[illegible]	[illegible]	[illegible]t
th[illegible]	[illegible]	[illegible]	[illegible]
unary-op v_A v_B	$v_A \leftarrow \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
binary-op v_A v_B v_C	$v_A \leftarrow v_B \otimes v_C$	$\tau(v_A) \leftarrow \tau(v_B) \cup \tau(v_C)$	Set v_A taint to v_B taint $\cup$ v_C taint
binary-op v_A v_B	$v_A \leftarrow v_A \otimes v_B$	$\tau(v_A) \leftarrow \tau(v_A) \cup \tau(v_B)$	Update v_A taint with v_B taint
binary-op v_A v_B C	$v_A \leftarrow v_B \otimes C$	$\tau(v_A) \leftarrow \tau(v_B)$	Set v_A taint to v_B taint
aput-op v_A v_B v_C	$v_B[v_C] \leftarrow v_A$	$\tau(v_B[\cdot]) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_A)$	Update array v_B taint with v_A taint
aget-op v_A v_B v_C	$v_A \leftarrow v_B[v_C]$	$\tau(v_A) \leftarrow \tau(v_B[\cdot]) \cup \tau(v_C)$	Set v_A taint to array and index taint
sput-op v_A f_B	$f_B \leftarrow v_A$	$\tau(f_B) \leftarrow \tau(v_A)$	Set field f_B taint to v_A taint
sget-op v_A f_B	$v_A \leftarrow f_B$	$\tau(v_A) \leftarrow \tau(f_B)$	Set v_A taint to field f_B taint
iput-op v_A v_B f_C	[illegible]	[illegible]	Set field f_C taint to v_A taint
ige[illegible]	[illegible]	[illegible]	[illegible]ject reference taint

$vA \leftarrow vB$ $vA \leftarrow vB$

$\tau(\quad) \quad \tau(\quad)$

$vA \leftarrow fB$ $vA \leftarrow fB$

$\tau(\quad) \quad \tau(\quad)$

Data flow control: dynamic analysis, – Limits

- **False negatives**

- **Management of data flows**
  - Explicit flows

```
y = x
```

- **Do not consider control flows**
  - Implicit flows

```
if (x)
  y = true
else
  y = false
```


IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Control dependencies attack

```
X = false
Y = false
char c[256];
If( gets(c) != user_contact )
  X = true;
else
  Y = true;
NetworkTransfer(x);
NetworkTransfer(y);
```

Data leakage

Our Research topics
– Solving “under-tainting” problem

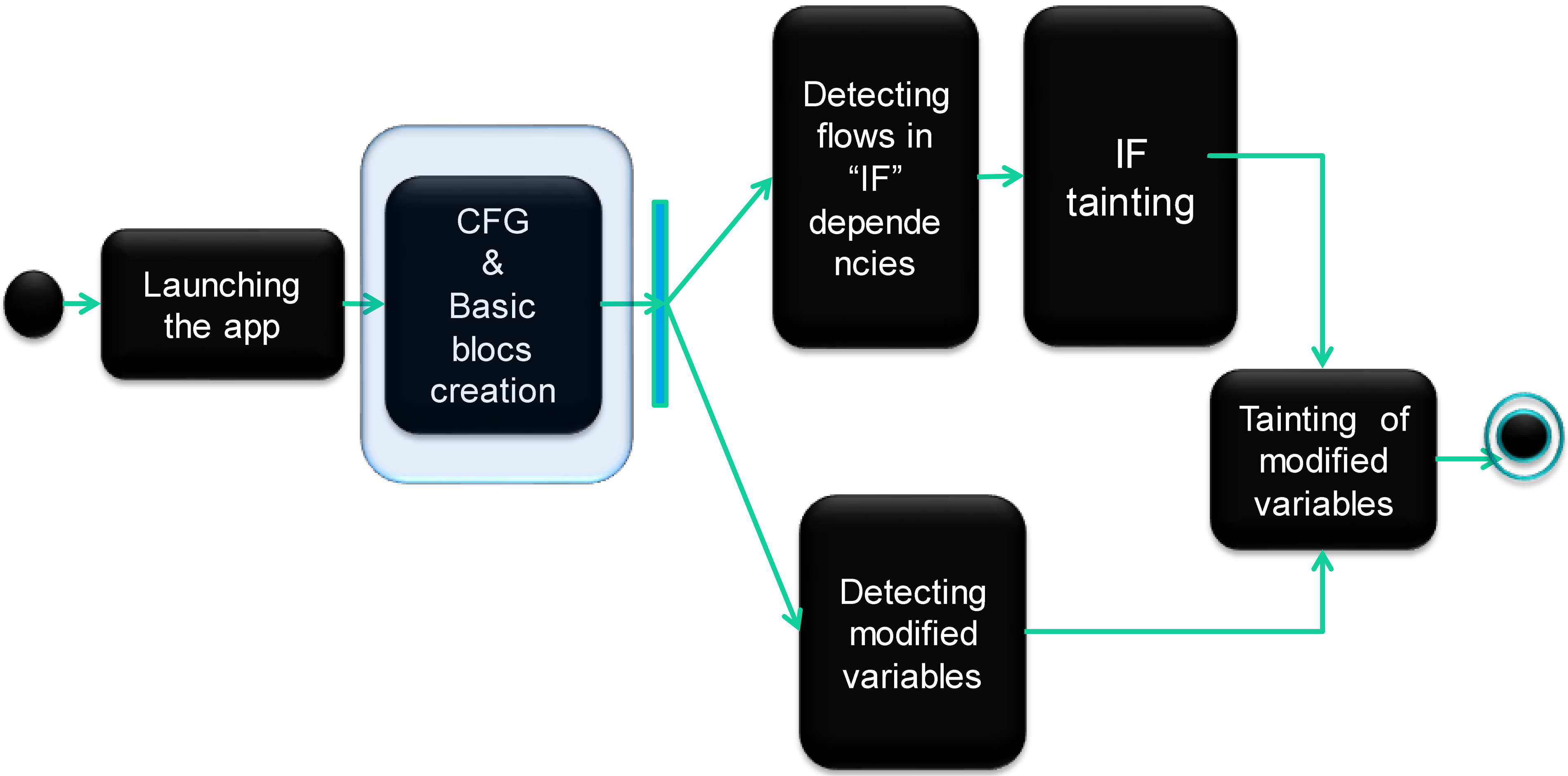
Leakage using flow control

- *Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Ana R. Cavalli: Detecting Control Flow in Smarphones: Combining Static and Dynamic Analyses. CSS 2012*

Code obfuscation

- *Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Ana R. Cavalli: Protection against Code Obfuscation Attacks Based on Control Dependencies in Android Systems. SERE 2014*

Solving “under-tainting” problem



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

New tainting propagation rules

$$(x \to y) \Rightarrow (Taint(y) \leftarrow Taint(x))$$

$$(x \leftarrow y) \Rightarrow (y \to x)$$

$$(Taint(x) \leftarrow Taint(y)) \wedge (Taint(x) \leftarrow Taint(z))$$
$$\Rightarrow (Taint(x) \leftarrow Taint(y) \oplus Taint(z))$$

$$\frac{Is\ modified(x) \wedge Dependency(x, condition) \wedge BranchTaken(br, conditional statement)}{Taint(x) \leftarrow Context\_Taint \oplus Taint(explicit flow statement)}$$

$$\frac{Is\ assigned(x, y) \wedge Dependency(x, condition) \wedge \neg BranchTaken(br, conditional statement)}{Taint(x) \leftarrow Taint(x) \oplus Context\_Taint}$$

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Obfuscated code

- **IMEI (International Mobile Equipment Identity)**

Dynamic analysis

```
1.X← User_Location
2.NetworkTransfer(X);
```

✓

```
1.X ← User_Location
2.for each x in X do
3.  For each symbol in
    AsciiTable do
4.    If(symbol = x then)
5.        Y ←Y + symbol
6.     end if
7.   end for
8.end for
9.NetworkTransfer(Y);
```

✗

Obfuscated code: Solved!

```
1.X ← User_Location
2.For each x in X do
3.   For each symbol in
  AsciiTable do
4.      If(symbol = x then)
5.           Y ←Y + symbol
6.      end if
7.   end for
8.end for

9.NetworkTransfer(Y);
```

✓

Our Research topics:
– Detecting flows in native codes

- **Instrumenting JNI code to avoid sensitive data leakage**
  - *Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Jean-Louis Lanet: Tracking Explicit and Control Flows in Java and Native Android Apps Code. ICISSP 2016*

```
package com.tuto.attackndk;
public class MainActivity extends
   Activity {
 static {
   System.loadLibrary("attackndk");}
public static native void
     invokeNativeFunction(String IMEI);
   @Override
    protected void onCreate(Bundle
   savedInstanceState) {

   super.onCreate(savedInstanceState);

   setContentView(R.layout.activity_main
   );
          String device_id =
   GetDeviceId();
      invokeNativeFunction(device_id);}}
```

Attack exploiting native code

```
#include <string.h>
#include <jni.h>
VoidJava_com_tuto_attackndk_MainActivity_invokeNati
   veFunction(JNIEnv* env, jobject thiz, jstring
   IMEI){
String Private_Data;
String Z;
strcpy(Private_Data, IMEI);
 for(int i = 0; i < sizeof(Private_Data); i++)
   {
      char s;
      sprintf(s, "%d", i);
   for(int j = 1; j < sizeof(TabAsc); j++)
       if(strcmp(s,TabAsc[j]) ==
   0)
            strcat(Z,TabAsc[j]);
    }
NetworkTransfer(Z);}
```

Native malicious function

Our Research topics
– Side channel attacks

- **Dealing with different side channel attacks**
  - *Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Jean-Louis Lanet, Routa Moussaileb: Detection of Side Channel Attacks Based on Data Tainting in Android Systems. SEC 2017*

$X \leftarrow Private\_Data$
for each $x \in X$ **do**
  $n \leftarrow CharToInt(x)$
  $StartTime \leftarrow ReadSystemTime()$
  $Wait(n)$
  $StopTime \leftarrow ReadSystemTime()$
  $y \leftarrow (StopTime - StartTime)$
  $Y \leftarrow Y + IntToChar(y)$
end for
$Send\_Network\_Data(Y)$

- **Timing attack example**
  - Enrich the tainting policy rules
  - The system clock is sensitive

Our Research topics – Hijacking attacks

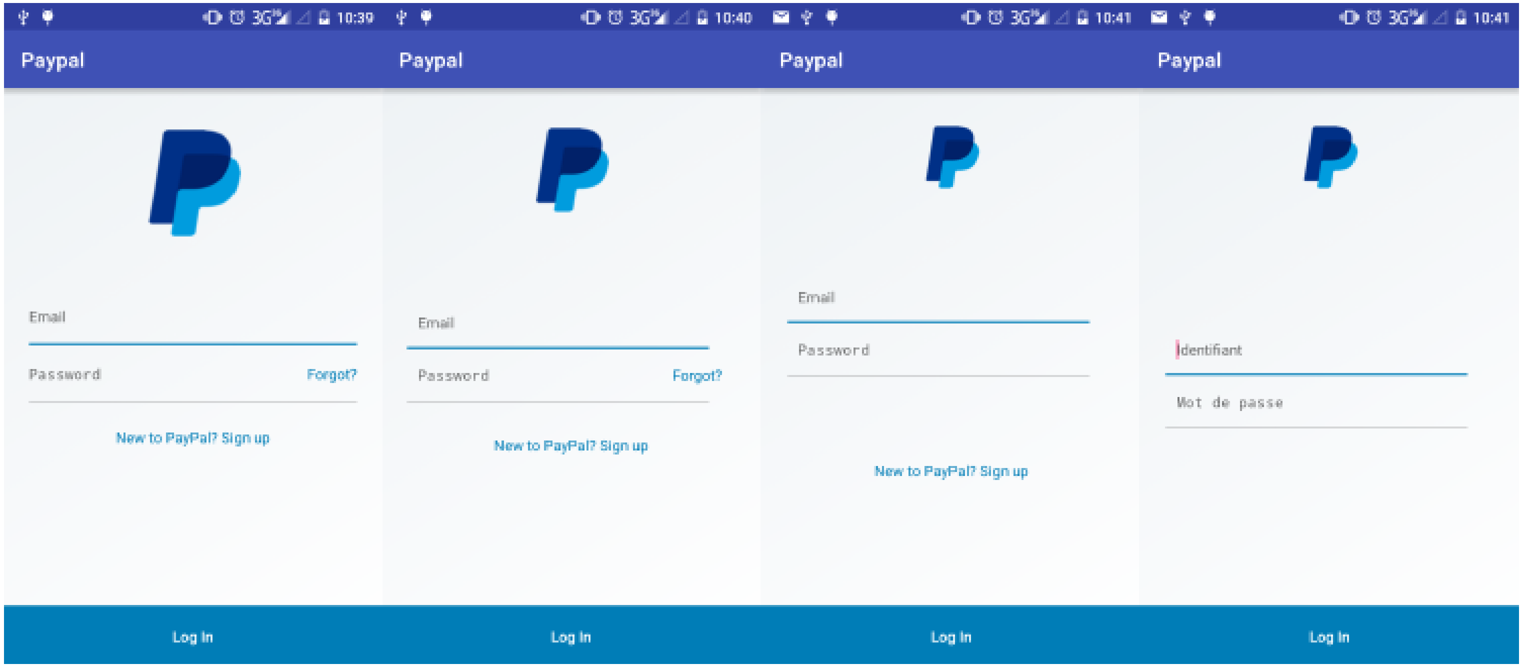
- **Detecting activity hijacking**
  - *Anis Bkakria, Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Jean-Louis Lanet. Experimenting similarity-based hijacking attacks detection and response in Android Systems ICISS'2017*
- **Android users Activities require to communicate sensitive data**
  - passwords, security codes, and credit card numbers) with applications
- **Hacker can launch hijacking attacks to compromise user's data confidentiality / privacy**
- **[Chen et al.] stealthily inject into the foreground a hijacking activity at the right timing and steal sensitive information in Android smartphones**

Our Research topics
– Hijacking attacks

- **The core security mechanisms of Android cannot detect activity hijacking**
- **Some not satisfactory existing work**
  - [Malisa et al.] and [Sun et al.] analyse application resource files (XML layout) to detect similarity of UI
- **Our proposal**
  - Modify the Android operating system
  - Extract and compare UI elements features of the legitimate and the hijacking interfaces
  - Use the indistinguishability level between the attack and legitimate Activities
  - Reacting: blocking or alerting

Our Research topics
– Hijacking attacks

- **False positives**
  - 4.2% in the case of partial indistinguishabilit
  - 10^{-3} % in the case of full indistinguishability
- **Performance:** 0.39% performance overhead on a CPU-bound micro-benchmark



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Conclusion

- **Malicious and behaviour in smartphone platforms has evolved significantly in the last decade**
  - Android particularly

- **It currently target Internet of Things devices**
  - Many open research problems in this context
    - Privacy of course,
    - But also Trust and Security that need to be revised

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Confidentiality violations detection in Android systems

Nora Cuppens
Directrice de Recherche

Colloque IMT'2017, November 10th