



# Architectural Risk Analysis for Android Applications

1

Habilitation Presentation, March 17th, 2021

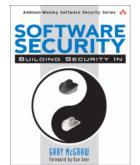
Dr. Karsten Sohr TZI – University of Bremen Center for Computing Technologies



#### **TZi** Software Security as an Own Discipline

- Usual security mechanisms such as firewalls, anti-virus software or intrusion detection systems are reactive
- Cause of many security problems: security holes *in software*
- McGraw: Trinity of trouble
  - 1. Increasing complexity (Windows 8 up to 80 Mio. lines of code?)
  - 2. Increasing connectivity (SOA, Internet of Things, industrial controllers...)
  - 3. Extensibility of systems (installation of apps, plugins for browsers)
- Tools and processes to improve software security
  - Security development lifecycle (SDL)





# **TZi** Code Review through Static Code Analysis

- Security analysis of the source code of applications
  - Detection of common programming bugs, such as buffer overflows, SQLinjection- and cross-site-scripting vulnerabilities
  - Automated analysis
- Use of compiler-construction techniques
  - Intermediate representation of the program e.g. by abstract syntax trees, static single assignment (SSA), System Dependence Graphs (SDGs)
  - Data- and control flow analyses
- False positives, false negatives
  - Non-decidability
- Commercial tools: Fortify SCA (for Java), IBM AppScan, Checkmarx, Veracode, Coverity Prevent (for C/C++ code)



# TZi Architectural Risk Analysis as Part of the SDL

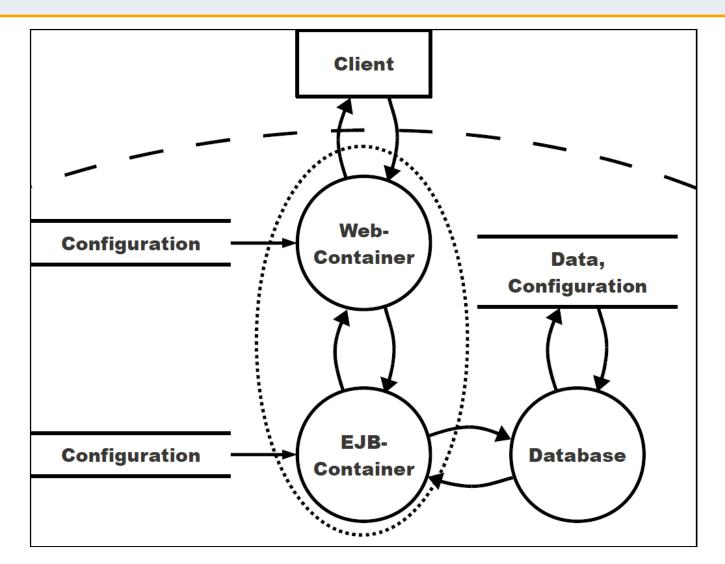
- Security analysis of the software architecture
  - At design time
  - Detection of basic security problems ("flaws" vs. "bugs")
  - Example of flaws: Missing encryption; only encryption, although integrity is required; authorization checks on client side; overprivilege; wrong usage of SW frameworks
- Several approaches, e.g.
  - Threat Modeling/STRIDE from Microsoft
  - Architectural risk analysis (ARA) from McGraw
- Core idea in such approaches:

Discussion of basic security aspects with the help of diagrams (forest-level overview)

4



#### **T** Threat Modeling with Dataflow Diagrams





# **TZi** Security Holes in Mobile Apps

- Overprivileged apps
- Confused-deputy problems
- Massive vulnerabilities in TLS-client implementation of apps
- Wrongly implemented encryption (e.g., insecure algorithms and crypto modes, insecure key generation)
- Injection of JavaScript code into apps with web functionality
- Faulty usage of software frameworks (Android framework)
- Weaknesses in systems consisting of a remote-control app and a backend



Insecure Usage of the Android Framework: Telekom Online Manager

Intent localIntent1 =

new

Intent("de.telekom.hotspot.intent.action.SMS\_STATUS");

7

localIntent1.putExtra("status",

CredSmsStatusType.SMS\_STATUS\_CREDENTIALS\_RECEIVED);

localIntent1.putExtra("username", str2);

localIntent1.putExtra("password", str3);

sendBroadcast(localIntent1);



# **TZi** Exported Content Provider: SAP Mobile Documents

```
<provider
android:name="com.sap.mcm.android.content.FileContentProvider"
android:exported="true"
android:authorities="com.sap.mcm.android.provider" />
```

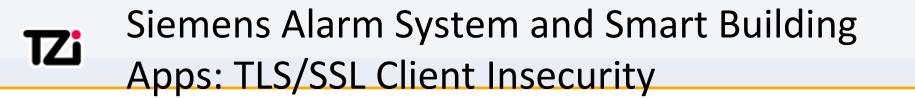
```
public File getFile(<u>Uri paramUri)</u>{
    McmDocument localMcmDocument = getDocument(paramUri);
    File localFile =
        new File(localMcmDocument.getUnencryptedPath());
    return localFile;
}
```

```
private void decryptNextDocument() {
```

McmDocument localMcmDocument =
 (McmDocument)this.documents.get(this.nextDocumentIndex);

8

localMcmDocument.copyUnencrypted(this);



```
public void onReceivedSslError(WebView
  paramWebView, SslErrorHandler
  paramSslErrorHandler, SslError
  paramSslError)
  {
    paramSslErrorHandler.proceed();
  }
```



# We need cost-efficient analysis methods and evaluation processes that assure that Android apps show an appropriate security level.



### **TZi** Research Project ZertApps

BMBF-funded project ZertApps: Certified security for mobile applications

#### Project partners:

- Universität Bremen
- Fraunhofer SIT (Prof. Dr. Eric Bodden)
- TU Darmstadt (Prof. Dr. Melanie Volkamer)
- OTARIS Interactive Service GmbH
- datenschutz cert GmbH
- SAP SE (Prof. Dr. Achim Brucker)







- Development of precise static security analyses which support the Android Framework (using the Soot Java analysis framework)
- If necessary, dynamic analysis to improve static analysis
- Interaction of several apps ( $\rightarrow$  confused deputy problem)
- Consideration of hybrid apps (apps with Java and web parts)
  - E.g., analysis of Cordova-based apps





- Comprehensible presentation of analysis results for different groups of users
  - Security administrators, evaluators, developers, users?
- Conception of a lightweight certification process / scheme
  - Low cost
  - Graded certification concept
- Tool-support for certification



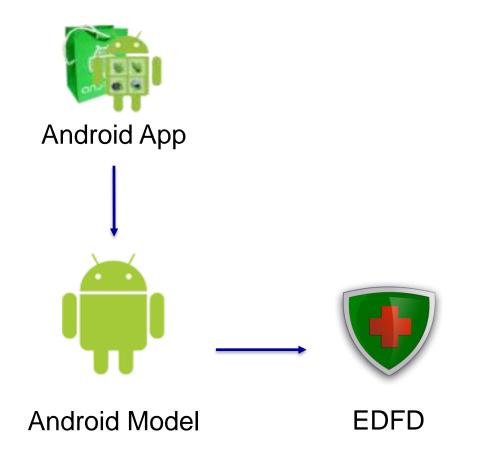
# **TZi** Our Contribution: Architectural Risk Analysis

- Extraction and security analysis of the software architectures of apps (or parts of the architectures)
- Reverse engineering of dataflow diagrams (DFDs) or extended dataflow diagrams (EDFDs) with the help of static analysis (with the help of Soot)
- Automated analysis of these extracted (E)DFDs against known architectural weaknesses (e.g., CWE entries)
- Conception and implementation within in the context of a dissertation at the AG Softwaretechnik (Bernhard Berger)



#### **TZi** Procedure: Android App to EDFD



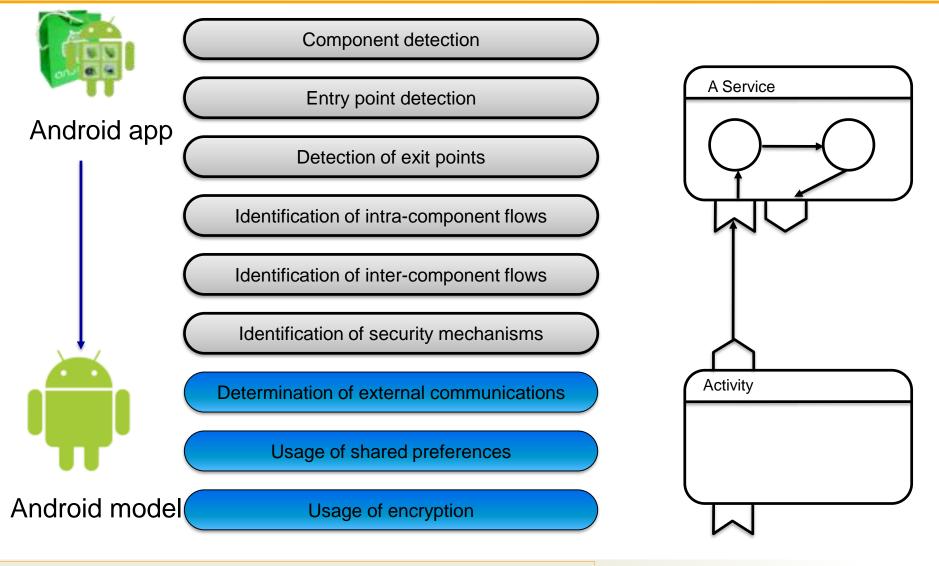




Empowering Digital Media Dr. Karsten Sohr, sohr@tzi.de

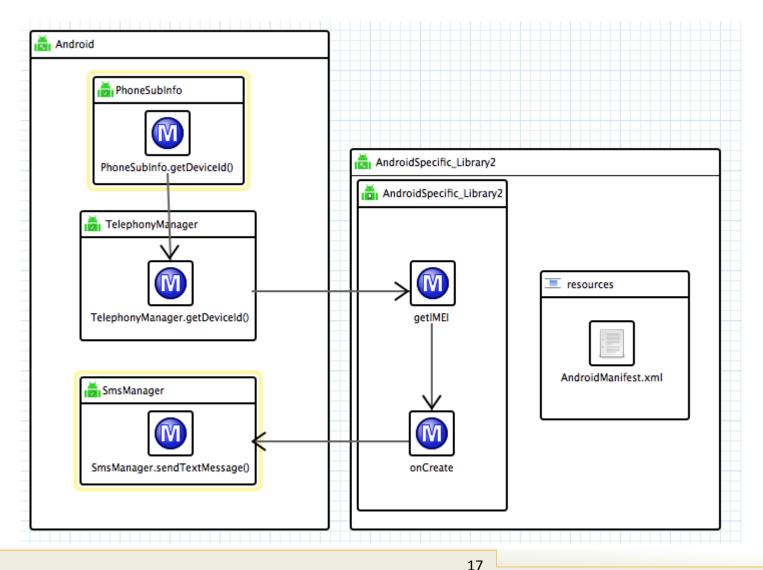
#### **TZi** Static Analysis Steps





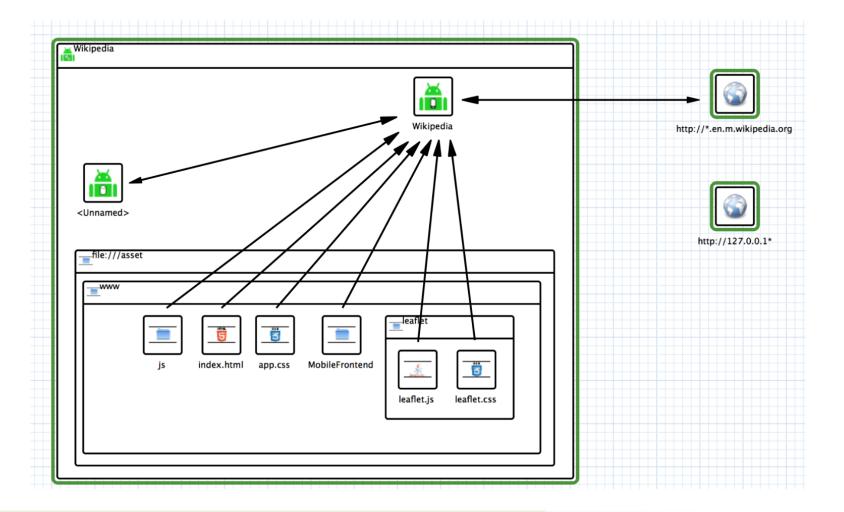
Empowering Digital Media Dr. Karsten Sohr, sohr@tzi.de

# **TZi** An Example Dataflow Diagram





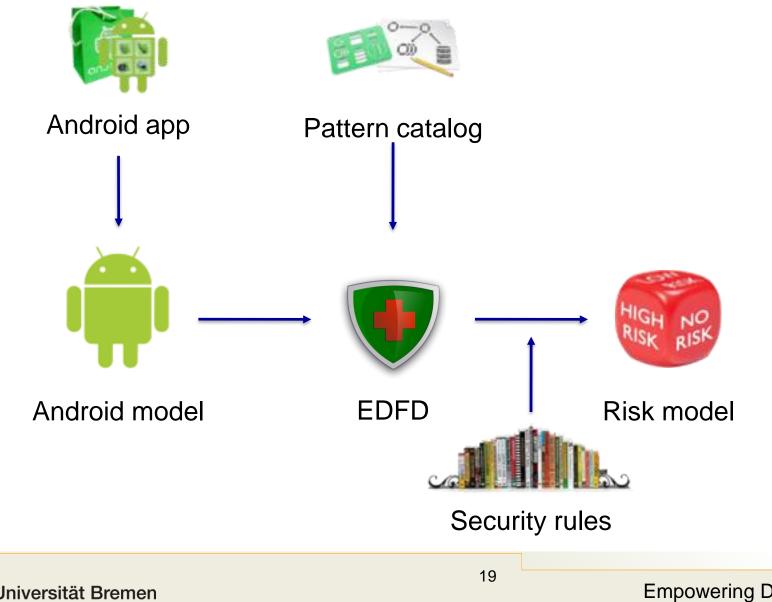
### **TZi** Another Dataflow Diagram: Hybrid App





Empowering Digital Media Dr. Karsten Sohr, sohr@tzi.de

# **TZi** Procedure: Creating Risk Models

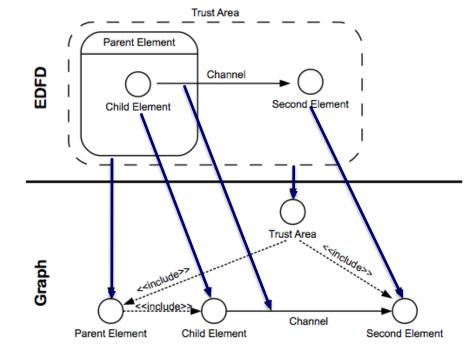


Empowering Digital Media Dr. Karsten Sohr, sohr@tzi.de

obile security

#### **TZi** Checking Security Rules





MATCH (source : Element) -[flow : Channel \*]-> (target : Element) WHERE flow.data.IsConfidential and not flow.IsEncrypted



Empowering Digital Media Dr. Karsten Sohr, sohr@tzi.de

# **TZi** Further Directions of this Work /1

- Applying this approach to Java/JEE applications and integrating it into a software certification platform
  - Project: CertifiedApplications (BMWi-funded, with datenschutz cert GmbH)
  - Paper: B. Berger, K. Sohr, R. Koschke. The Architectural Security Tool Suite ArchSec, 19th IEEE International Working Conference on Source Code Analysis and Manipulation, Cleveland, Ohio, 2019.
- Combined analysis: Android app and backend application
  - Constructing a common DFD (Android app + backend DFD)
  - Connection by using the external interface of the sever (e.g., SAP Mobile Documents app and its underlying content management system)
- Analysis of Android apps containing native code
  - Analysis supporting Java as well as C/C++ code
  - E.g., Java-based Android app with Qt parts (e.g., AusweisApp2 for Android)



# **TZi** Further Directions of this Work /2

- Applying this approach to other Java-based software framework: E.g., Spring, Apache Shiro
- Applying this approach to other programming languages
  - C/C++:

IoT applications or apps with C/C++ parts

Basic analysis infrastructure: LLVM compiler infrastructure (similar to Soot and WALA, but for C/C+)

• Microsoft C#:

.NET framework, in practice often used, well-documented and clearly defined API

Missing basic analysis infrastructure for C# (maybe, Bauhaus tool-suite)



# **1** Other Approaches: Slicing-Based Code Review

- Backward Slicing: Static program analysis technique that calculates all statements influencing a given seed statement (via control and data dependency): Which statements influence the seed?
- Idea: Use security-critical API calls as slicing seed statements and calculate backward slice from them
- Analysts can use these slices within code review tasks
- Using the WALA program analysis framework for this purpose
- Paper: Mustafa, T., Sohr, K. Understanding the implemented access control policy of Android system services with slicing and extended static checking. *Int. J. Inf. Secur.* 14, 347–366 (2015).



# **TZi** Application of the Slicing-Based Approach

- Android System Services (part of the Android platform)
  - Automated extraction of the implemented access control policy of Android system services by slicing
  - Seeds: E.g., checkCallingPermission, checkPermission
- JEE applications, programmatic access control: isCallerInRole
- Java crypto as wells as Java communications security (e.g., Cipher.doFinal, URLConnection.connect)
  - Has the crypto code been implemented correctly?
  - Have communications been secured appropriately?
- To improve results of data dependencies: Specific pointer analyses had to be implemented (or available points-to information had to be used)
- Slicing-based analysis tool available:

https://github.com/BitFlipp3r/AndroidSlicer-Evaluation



# **TZi** Slicing-Tool for Android: User Interface

<complex-block>And and a state of the stat</complex-block>		× +			—		٥
<section-header></section-header>	C 0	🔽 🖸 localhost:8080/slices/5f2ae566411f641bc8340374/view 50% 🗰 🖾 🔍 Suchen		$\overline{\mathbf{h}}$	lii\		(
<code-block></code-block>	<b>cer</b> v1.0.0		倄 Home 🛛 🐻 Slic	oes 🗱	📽 Settings	💄 Adır	nini
<code-block></code-block>						_	
<code-block></code-block>	C	lica 5f2aa566/11f6/1bc83/037/					
<page-header></page-header>							
<pre>bit product in the product of t</pre>							
<pre>SedSences The set of the set</pre>			e prepareForAuthentication				
		artPreparedClient addLockoutResetCallback rename resetTimeout enumerate enroll removeClientActiveCallback hasEnrolledTingerprints					
<pre>state description: The state description of the state description</pre>				_			
Show Diff       Wew Sde by Sde         # figure interview in the interview inter			gPermission enforcePermission	n			
Show Diff       Wew Sde by Sde         # figure interview in the interview inter							
Shou Dif			Open in IDE Do	wnload			
<pre>     @Override // Binder call     @Override // Binder call     @Override // Binder call     # f (correspondency */,     # f (corresponden</pre>							
<pre></pre>							
91     UserHandle.getCallingUserTd())) {     1000000000000000000000000000000000000			Service and the service of the servi				
<pre>91   UserHandLe_getCallingUserTd())) {</pre>		<pre>88 public List<fingerprint> getEnrolledFingerprints(int userId, String opPackageName) {</fingerprint></pre>	Plantanae. Plantanae.				
<pre>91       userHandle.getCallingUserId()) {</pre>			Maddiana				
93       }         94       }         95       @Override // Binder call         96       public bolean hashnolledingerprints(int userid, String opPackageName) {         96       if (frankselometric(opPackageName, false /* foregroundOnly */,         99       if (sinder.getCallingUid(), Binder.getCallingPid(),         101       return false;         102       ;         103       ;         104       return false;         105       goverride // Binder call         106       public void resetTimeout(byte [] token) {         107       checkPermission(RESET_FINGERPRINT_LOCKOUT);         108       ;         109       ;         101       public void resetTimeout(byte [] token) {         106       public void resetTimeout(byte (] token) {         107       checkPermission(RESET_FINGERPRINT_LOCKOUT);         108       ;         109       ;         108       ;         109       ;         109       ;         100       ;         101       public bolean isClientActive() {         Source File:       ;         1       ;         2       ;			<ul> <li>Selections</li> </ul>	-			
93       }         94       }         95       @Override // Binder call         97       public boolean hastnolled/ingerprints(int userid, String opPackageName) {         98       if (translesionetric/opPackageName, false /* foregroundOnly */,         99       isinder.getCallingUid(), Binder.getCallingDid(),         100       return false;         101       return false;         102       ;         103       ;         104       return false;         105       public void resetTimeout(byte [] token }         106       public void resetTimeout(byte [] token }         107       checkPermission(RESET_FINGERPRINT_LOCKOUT);         108       ;         109       @Override         101       public boolean isClientActive() {         Source File:       1         2       * coppright (c) 2814 The Android Open Source Project		91 UserHandle getCallingUserId())) {	Contractor **	-			
95       0         95       0         96       0         97       public boolean hastmolledingerprints(int userld, String opPackagellame) {         98       if (lcanUseBioneticlopPackagellame, false /* foregroundonly */, sinder.getCallingUserld(), sinder.getCallingUserld(), userHandle.getCallingUserld()) {         100		92 return Collections.emptyList();	1 States with the				
96       @Override // Binder call       "Magnature"         97       public boolean hasformulation userid, String opPackageName) {       "Magnature"         98       if (icaluseBiometric(opPackageName, faile /* foregroundOnly */,       "Magnature"         99       if (icaluseBiometric(opPackageName, faile /* foregroundOnly */,       "Magnature"         99       if (icaluseBiometric(opPackageName) {       "Magnature"         99       if (icaluseBiometric(opPackageName, faile /* foregroundOnly */,       "Magnature"         99       if (icaluseBiometric(opPackageName) {       "Magnature"         99       if (icaluseBiometric(opPackageName) {       "Magnature"         99       if (icaluseBiometric(opPackageName) {       "Magnature"         90       if (icaluseBiometric(opPackageName) {       "Magnature"         91       public void resetTimeou(byte [] token) {       "Magnature"         92       icaluseBiometric(opPackageName) {       "Magnature"         93       if (icaluseBiometric(icaluseBiometric(icaluseBiometric(icaluseBiometric(ical		<pre>92 return Collections.emptyList(); 93 }</pre>	CONSIDER THE PARTY OF THE PARTY				
99       Binder,getCallingUid(), Sinder.getCallingPid(),         100       UserHandle.getCallingUserId())) {         101       return fals;         102       }         103       gOverride // Binder call         104       gOverride // Binder call         105       public void resetTimeout(byte [] token) {         106       public void resetTimeout(byte [] token) {         107       heckPermission(RESET_FINGERPRINT_LOCKOUT);         108       }         109       gOverride         109       goverride         109       goverride         109       goverride         100       goverride         101       goverride         102       /*         103       goverride         104       goverride         105       public boolean isClientActive() {         Source File:       interaction         1       /*         1       /*         1       /*         1       /*         1       /*         1       /*         1       /*         1       /*         1       /*         1 </td <td></td> <td><pre>92 return Collections.emptyList(); 93 } 94 }</pre></td> <td>Telephone and</td> <td></td> <td></td> <td></td> <td></td>		<pre>92 return Collections.emptyList(); 93 } 94 }</pre>	Telephone and				
99       Binder,getCallingUid(), Sinder.getCallingPid(),         100       UserHandle.getCallingUserId())) {         110       return fals;         120       }         121       }         122       }         123       }         124       }         125       }         126       @Override // Binder call         126       public void resetTimeout(byte [] token) {         127       heckPermission(RESET_FINGERPRINT_LOCKOUT);         128       }         129       @Override         120       @Override         120       @Override         120       Source File:         12       * Copyright (C) 2014 The Android Open Source Project		<pre>92 return Collections.emptyList(); 93 } 94 } 95  96 @Override // Binder call</pre>					
100       UserHandle.getCallingUserId())) {         101       return false;         102       }         103       }         104		<pre>92 return Collections.emptyList(); 93 } 94 } 95 96 @Overnide // Binder call 97 public boolean hasEnrolledFingerprints(int userId, String opPackageName) {</pre>					
101       return false;         102       }         103       }         104       @Override // Binder call         106       public void resetTimeout(byte [] token) {         107       checkeremission(RESET_FINGERPRINT_LOCKOUT);         108       }         109       @Override         110       public boolean isclientActive() {         Source File:       1         1       *         2       * Copyright (C) 2014 The Android Open Source Project		<pre>22 return Collections.emptyList(); 33 } 34 } 35 36 @Override // Binder call 37 public booleam hasEnnolledFingerprints(int userId, String opPackageName) { 38 if (IcaniseEstometric(opPackageName, false /* foregroundOnly */, 38 if (IcaniseEstometric(opPackageName, false /* foregroundOnly */, 39 } 30 } 31 } 32 } 33 } 34 } 35 } 36 } 37 } 37 } 38 } 39 } 39 } 30 } 30 } 30 } 30 } 30 } 30 } 30 } 31 } 32 } 32 } 33 } 34 } 35 } 36 } 37 ] 37 } 37 ] 37 ] 37</pre>					
<pre>183   } 194 194 195 @Override // Binder call 195 @Override // Binder call 196   public void resetTimeout(byte [] token) { 197   checkermission(RESE_FINGERPRINT_LOCKOUT); 198   } 199 199 19 10 @Override 110 public boolean isclientActive() { Source File: 1 /* copyright (C) 2014 The Android Open Source Project 1 /* copyright (C) 2014 The Android Open Source Proj</pre>		<pre>92 return Collections.emptyList(); 93 } 94 } 95  96 @Overnide // Binder call 97 public boolean hasEnrolledFingerprints(int userId, String opPackageName) { 98 if (teanUseBiometric(opPackageName, failse /* foregroundDn)*/, 99 Binder.getCallinguid(), Binder.getCalling</pre>					
194		<pre>92 return Collections.emptyList(); 93 94 94 95 95 96 @Override // Binder call 97 public boolean hasEnnolLedFingerprints(int userId, String opPackageName) { 98 if (IcanUseESiometric(opPackageName, false /* foregroundOnly */, 99 Binder.getCallingUid(), Binder.getCallingUid(), 100 UserHandle.getCallingUiserId())) { 101 return false; 102 // 1</pre>					
105       @Override // Binder call         106       public void resetTimeout(tyte [] token) {         107       checkPermission(RESET_FINGERRINT_LOCKOUT);         108       }         109       @Override         110       gOverride         111       public boolean isClientActive() {         Source File:         1       /*         2       * Copyright (C) 2014 The Android Open Source Project		<pre>22 return Collections.emptyList(); 33 } 94 } 95 95 96 @Ovenride // Binder call 97 public boolean hasEnonlledFingerprints(int userId, String opPackageName) { 98 if (tansUseStometric(opPackageName, false /* foregroundonly */, 99 Binder.getCallingUid(), Binder.getCallingPid(), 108 UserHandle.getCallingUid(), Binder.getCallingPid(), 109 return false;</pre>					
187       i checkPermission(RESET_FINGERPRINT_LOCKOUT);         188       }         199       @Overnide         111       public boolean isClientActive() {         Source File:         Improve file:         2       * Copyright (C) 2014 The Android Open Source Project         Improve file:		92     return Collections.emptyList();       93     }       94     }       95     @Override // Binder call       96     @Override // Binder call       97     public boolean hastnolledFingerprints(int userId, String opPackageName) {       98     if (IcaniseEstometric)opPackageName) {       99     Binder.getCallingUid(), Binder.getCallingPid(),       101     return false;       102     }					
108     }       109     @Overnide       111     public boolean isClientActive() {   Source File:       1     /*       2     * copyright (C) 2014 The Android Open Source Project		<pre>22 / return Collections.emptyList(); 33 } 34 } 35 66 @Overnide // Binder call 77 public boolean hastnrolledFingerprints(int userId, String opPackageName) { 38 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 39 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 30 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 30 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 31 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 32 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 33 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 34 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 35 if (ta</pre>					
109 110     @Override public bolean isClientActive() {       Source File:       1       2     * Copyright (C) 2814 The Android Open Source Project		<pre>22 return Collections.emptyList(); 33 } 44 } 55 66 @Override // Binder call 97 public boolean hasEnrolledFingerprints(int userId, String opPackageName) { 98 if (IcanUseEiometric(opPackageName, false /* foregroundOnly */, 99 Binder.getCallingUld(), Binder.getCallingPid(), 108 UserHandle.getCallingUserId())) { 109 return false; 102 } 103 } 104 105 @Override // Binder call 106 @Override // Binder call 106 public void resetTimeout(byte [] token) { 107 // Content of the setTimeout(byte [] token) { 108 // Content of the setTimeout(byte [] token) { 109 // Content of the setTimeout(byte [] token) { 100 // Content of the setTimeout(byte [] token) { 100 // Content of the setTimeout(byte [] token) { 101 // Content of the setTimeout(byte [] token) { 102 // Content of the setTimeout(byte [] token) { 103 // Content of the setTimeout(byte [] token) { 104 // Content of the setTimeout(byte [] token) { 105 // Content of the setTimeout(byte [] token] { 105 // Content of the setTimeout(byte [] toke</pre>					
111     public boolean isClientActive() {       Source File:		<pre>22 / return Collections.emptyList(); 33 } 34 } 35 46 // Sinder call 57 public boolean hasIncolledFingerprints(int userId, String opPackageName) { 58 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 59 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 59 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 50 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 50 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 51 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 52 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 53 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 54 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 55 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 56 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 56 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 57 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 58 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 58 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 59 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 50 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 50 if (tanusEstionetric(opPackageName, false /* foregroundonly */, 51 if (tanusEstione</pre>					
Source File:		<pre>22 / return Collections.emptyList(); 33 } 34 } 35 35 46 @Overnide // Binder call 36 @Overnide // Binder call 37 public boolean hastmolledFingerprints(int userId, String opPackageName) { 38 if (tanUseStometric(opPackageName, false /* foregroundonly */, 39 if (tanUseStometric(opPackageName, false /* foregroundonly */, 30 if (tanUseStometric(opPackageName, false /* foregroundonly */, 31 if (tanUseStometric(opPackageName, false)) if (tanUse</pre>					
1 /*		<pre>22 / return Collections.emptyList(); 33 } 44 } 55 / 56 @Override // Binder call 57 public bolean hashnolladfingerprints(int userId, String opPackageName) { 58 if (lanUseBiometric(opPackageName, false /* foregroundOnly */, 59 linder.getCallingUid(), Binder.getCallingPid(), 50 // UserHandle.getCallingUid(), Binder.getCallingPid(), 50 // UserHandle.getCallingUid(), Binder.getCallingPid(), 51 // UserHandle.getCallingUid(), Binder.getCallingPid(), 52 // UserHandle.getCallingUid(), Binder.getCallingPid(), 53 // UserHandle.getCallingUid(), 54 // UserHandle.getCallingUid(), 55 // UserHandle.getCallingUid(), 56 // UserHandle.getCallingUid(), 57 // UserHandle.getCallingUid(), 58 // UserHandle.getCallingUid(), 59 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 51 // UserHandle.getCallingUid(), 52 // UserHandle.getCallingUid(), 53 // UserHandle.getCallingUid(), 54 // UserHandle.getCallingUid(), 55 // UserHandle.getCallingUid(), 56 // UserHandle.getCallingUid(), 57 // UserHandle.ge</pre>					
2 * Copyright (C) 2014 The Android Open Source Project		<pre>22 / return Collections.emptyList(); 33 } 44 } 55 / 56 @Override // Binder call 57 public bolean hashnolladfingerprints(int userId, String opPackageName) { 58 if (lanUseBiometric(opPackageName, false /* foregroundOnly */, 59 linder.getCallingUid(), Binder.getCallingPid(), 50 // UserHandle.getCallingUid(), Binder.getCallingPid(), 50 // UserHandle.getCallingUid(), Binder.getCallingPid(), 51 // UserHandle.getCallingUid(), Binder.getCallingPid(), 52 // UserHandle.getCallingUid(), Binder.getCallingPid(), 53 // UserHandle.getCallingUid(), 54 // UserHandle.getCallingUid(), 55 // UserHandle.getCallingUid(), 56 // UserHandle.getCallingUid(), 57 // UserHandle.getCallingUid(), 58 // UserHandle.getCallingUid(), 59 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 50 // UserHandle.getCallingUid(), 51 // UserHandle.getCallingUid(), 52 // UserHandle.getCallingUid(), 53 // UserHandle.getCallingUid(), 54 // UserHandle.getCallingUid(), 55 // UserHandle.getCallingUid(), 56 // UserHandle.getCallingUid(), 57 // UserHandle.ge</pre>					
z copyright (c) zora nie androza open boarce project		<pre>22 / return Collections.emptyList(); 34 } 45 } 56 @Override // Binder call 57 public boolean hasfnonlledFingerprints(int userId, String opPackageName) { 58 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 59 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 50 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 51 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 52 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 53 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 54 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 55 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 56 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 56 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 57 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 58 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 59 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 50 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 51 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 51 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 51 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 52 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 53 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 54 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 54 if (IcanuseBiometric(opPackageName, false /* foregroundonly */, 55 if (IcanuseBiometric(opPackageName, false /*</pre>					
		<pre>22         return Collections.emptyList(); 33   } 44 } 55   60 @Overnide // Binder call 97 public boolean hasEnolledFingerprints(int userId, String opPackageName) { 98   If (tanNesBeSometric(opPackageName, false /* foregroundon) */, 99   Binder.getCallingUid(), Binder.getCallingPid(), 99   UserHandle.getCallingUid(), Binder.getCallingPid(), 100   vserHandle.getCallingUid(), Binder.getCallingPid(), 101   return false; 102   } 103   } 104   Overnide // Binder call 105   @Overnide // Binder call 106   public void resetTimeout(byte [] token) { 107   checkPermission(RESET_FINGERPRINT_LOCKOUT); 108   } 109   @Overnide // Binder.call 100   @Overnide // Binder.call 100   Overnide // Binder.call 100   Overnide // Binder.call 101   Overnide // Binder.call 102   /*</pre>					



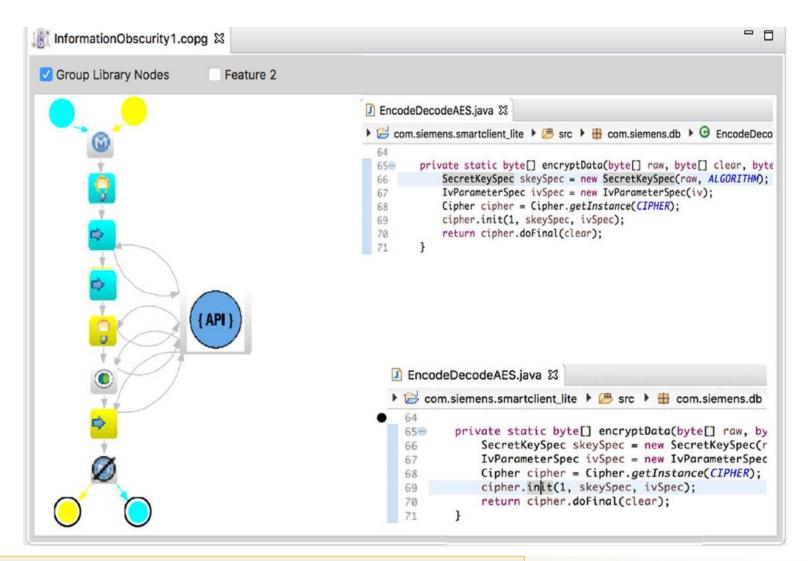


# Other Approaches: Extracting Security Patterns from Java Code

- Use static program analysis for extracting implemented security patterns from Java code
- Informally: Security patterns counterparts to design patterns
- Examples of security patterns:
  - Secure Channel
  - Secure Storage
  - Authenticator
- Idea: Represent manifestations of security patterns in code as Connected
   Object Process Graphs (COPGs)
  - Underlying observation: Security patterns are often represented in code by connected Java objects, e.g. Cipher, SecretKey, SecretKeySpec objects
- Paper: Bunke, M., Sohr, K. Towards supporting software assurance assessments by detecting security patterns. *Software Quality Journal* 28, 1711–1753 (2020)



### **TZi** An Example COPG





27

# **TZi** User Studies

- Still missing for all approaches: User studies
- Who are the addressed users of such tools?
  - Security analysts and evaluators
  - Architects
  - Developers
- Find adequate user groups and users (not that easy)
- Define user experiments
  - Controlled experiments
  - Baseline: Manual code review, bug finders for review
- Not necessary to come up with a complete tool
  - Demonstrate that *representations help (or not)*



# **TZi** Summary & Outlook

- Software security becomes more relevant
  - Mobile apps, Internet of Things, industrial controllers, ...
- Systematic and cost-efficient processes for software security are needed
- Especially relevant: security of apps
- Tool support
- Static (and dynamic) code analyses for the extraction and validation of the implemented security architecture
- Comprehensive user experiments/studies necessary to demonstrate efficacy/usability of the developed static analysis tools

