Is There Value in Reasoning about Security at the Architectural Level: a Comparative Evaluation

Ebrahim Khalaj

Radu Vanciu

Marwan Abi-Antoun

Department of Computer Science, Wayne State University

*This work was supported in part by the Army Research Office under Award No. W911NF-09-1-0273.

Finding security vulnerabilities that are closer to **architectural flaws** is harder

Architectural flaw
e.g., missing authentication

Coding bug
e.g., hard-coded password

Approaches make tradeoffs

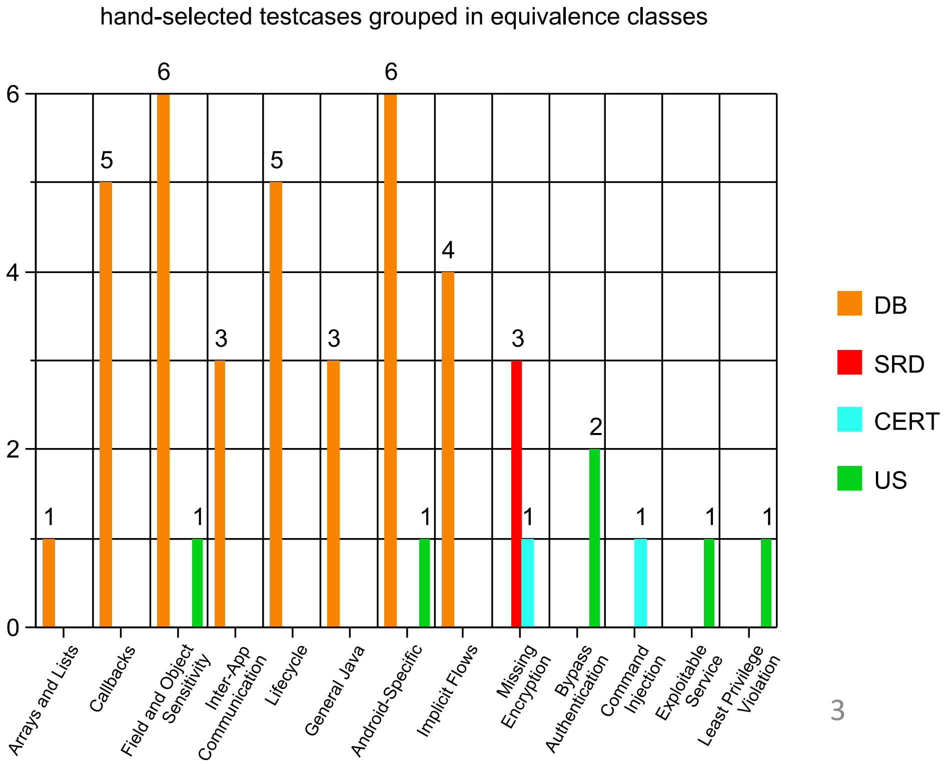
Sound and possibly less precise	Unsound and possibly more precise
Analyst-assisted approach	More automated approach;
Special purpose constraints	General purpose constraints
Separate extraction and constraints	Combined extraction and constraints
High-level representation of the system	Code-oriented view of the system
…	…

Comparing approaches that find architectural flaws using a benchmark

- Some Common Weakness Enumerations (CWE) related to architectural flaws without corresponding testcases*
  - CWE-325: Missing Required Cryptographic Step (34 testcases)
  - CWE-311: Missing Encryption of Sensitive Data (no testcases)

- ScoriaBench
  - 43 hand-selected testcases
  - Android and Java applications
  - 13 different equivalence classes

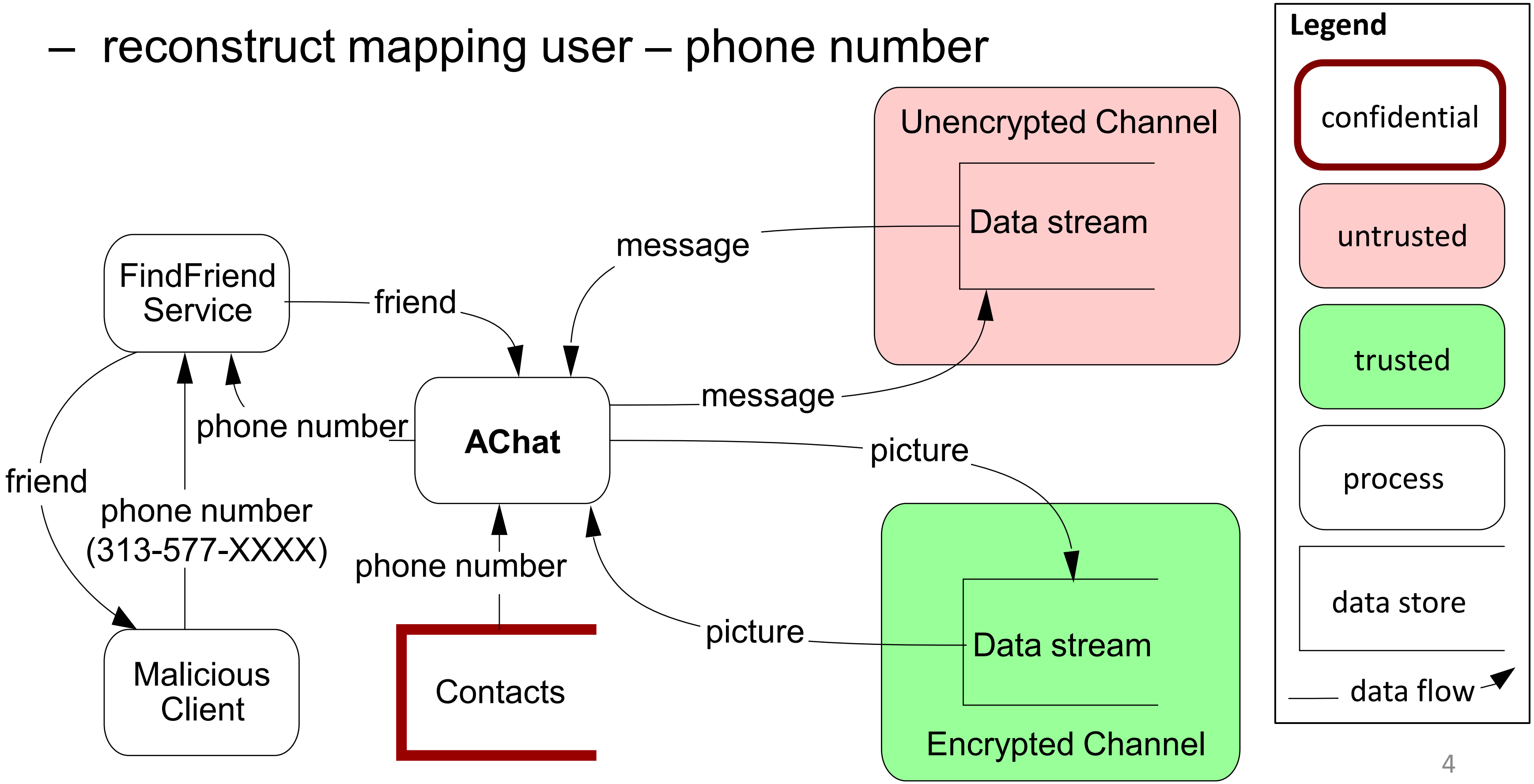
- Selected test cases from
  - DroidBench(DB)
  - SAMATE Reference Dataset (SRD)
  - CERT rules examples
  - Designed by us (US)



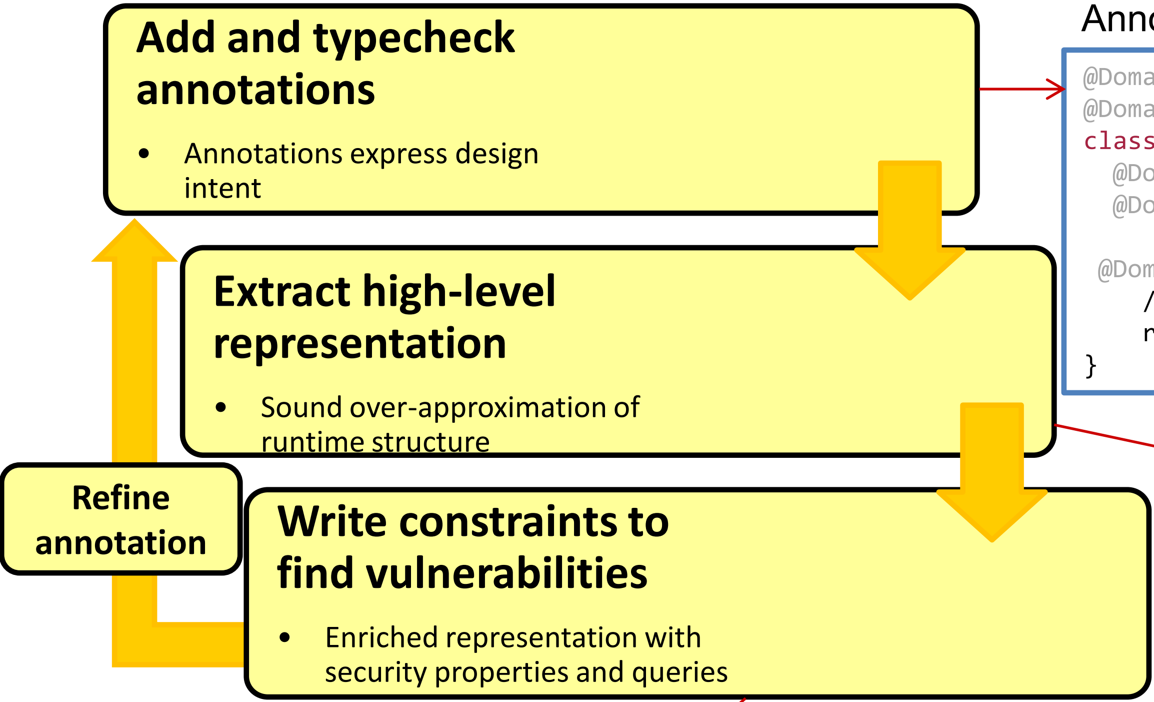
*in SRD Juliet Test Suite for Java

Exploitable FindFriend Service

- No transitive information flow from Contacts to Client
- Brute force attack
  - reconstruct mapping user – phone number



Scoria process [Vanciu and Abi-Antoun, ASE'2013]



Add and typecheck annotations

- Annotations express design intent

Extract high-level representation

- Sound over-approximation of runtime structure

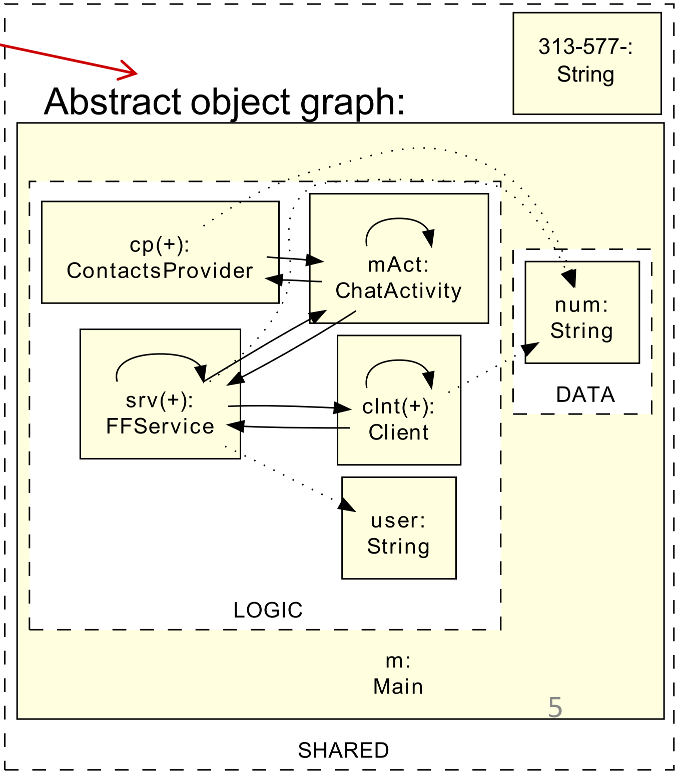
Write constraints to find vulnerabilities

- Enriched representation with security properties and queries

Refine annotation

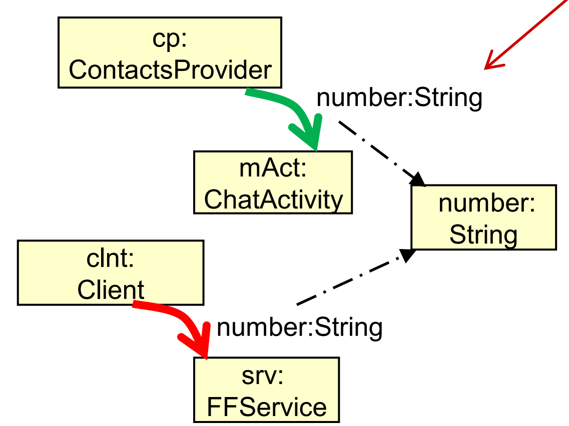
Annotated code:

```
@DomainParams({"U","L","D"})
@Domains({"SLIST", "owned"})
class FFService {
  @Domain("owned<D,L>")HashMap<String,String> map = ...
  @Domain("SLIST<L>")List<String>
              findByNumber(@Domain("D")String num) {
 @Domain("SLIST<L>") List<String> uNames = ...;
    //search for the number in the map
    return uNames;
}
```



Abstract object graph:

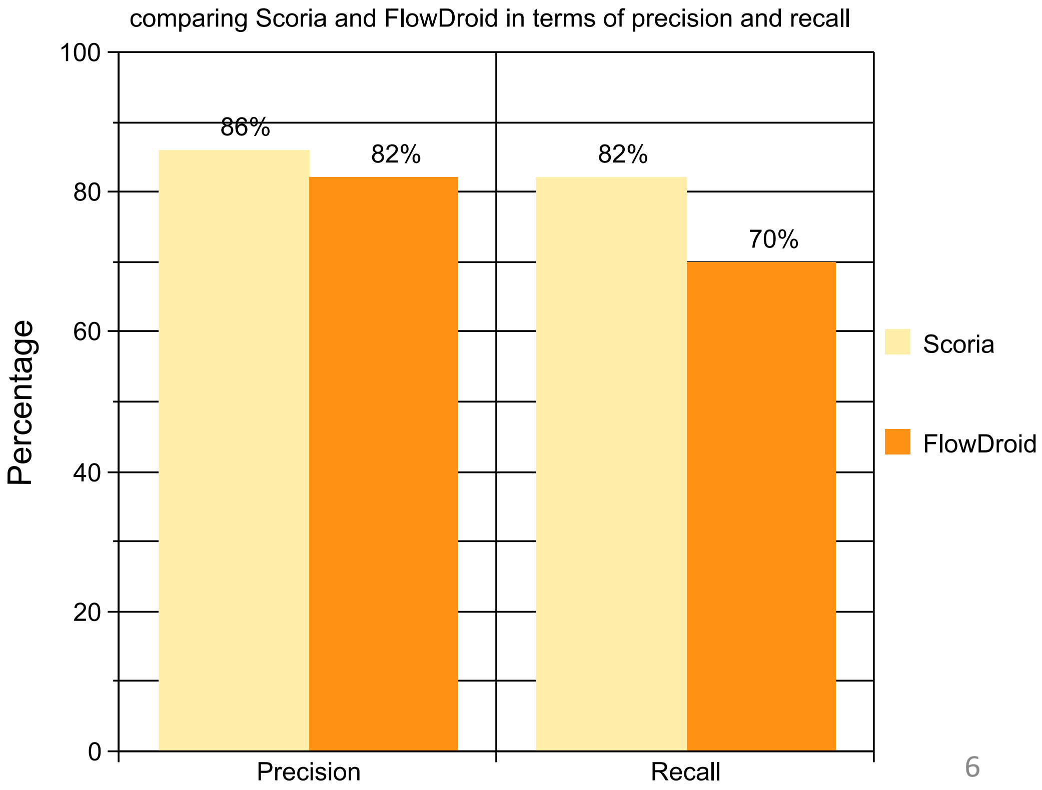
Constraint:



Results

- Compare in terms of precision and recall
  - Scoria
  - FlowDroid [Arzt et al., PLDI'2014]

Precision = (TP)/(TP+FP)
Recall = (TP)/(TP+FN)


comparing Scoria and FlowDroid in terms of precision and recall
Percentage
Precision
Recall
Scoria
FlowDroid