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 **{mekhalaj, radu, mabiantoun}@wayne.edu**

Method

We want to **compare** tools that find security vulnerabilities. We propose a **benchmark** with **hand-selected** testcases.

Different resources for testcases used to build ScoriaBench:

- DroidBench(DB)
- SAMATE Reference Dataset (SRD)
  - Based on Common Weakness Enumeration
- CERT rules examples
- Designed by us (US)

CWE-592 Authentication Bypass Issues

The software does not properly perform authentication, allowing it to be bypassed through various methods.

CWE-290 Authentication Bypass by Spoofing

This attack-focused weakness is caused by improperly implemented authentication schemes that are subject to spoofing attacks.

CWE-302 Authentication Bypass by Assumed-Immutable Data

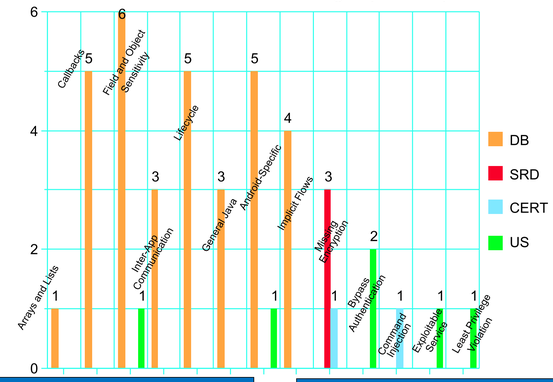
The authentication scheme or implementation uses key data elements that are assumed to be immutable, but can be controlled or modified by the attacker.

ScoriaBench www.cs.wayne.edu/~mabianto/sb

A benchmark with testcases that are grouped in different **equivalence classes**. Some testcases focus on **architectural flaws.**

Architectural flaw e.g., missing authentication — **Coding bug** e.g., hard-coded password

number of testcases from different resources grouped in equivalence classes

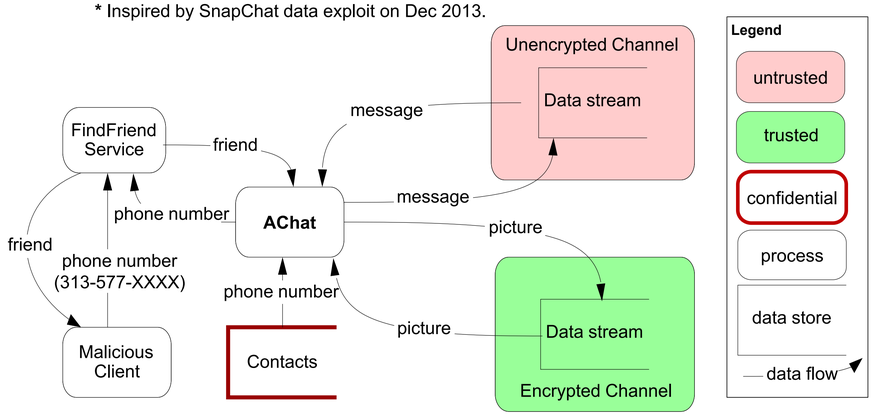


Example

Our goal is to find security vulnerabilities such as **information disclosure**.

AChat: A chat application that discloses **confidential information** of its users to a **malicious client**. This example represents an **exploitable service**.

* Inspired by SnapChat data exploit on Dec 2013.



Comparison Metrics

We compare approaches in terms of:

-True Positives (**TP**): a real vulnerability reported by the tool. Higher is better;

-False Positives (**FP**): a vulnerability that does not exist but is reported by the tool. Lower is better;

-False Negatives (**FN**): a real vulnerability that is missed by the tool. Lower is better.

Then we use them calculate comparison **metrics**:

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

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

Add and typecheck annotations
- Annotations consistent with code
- 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

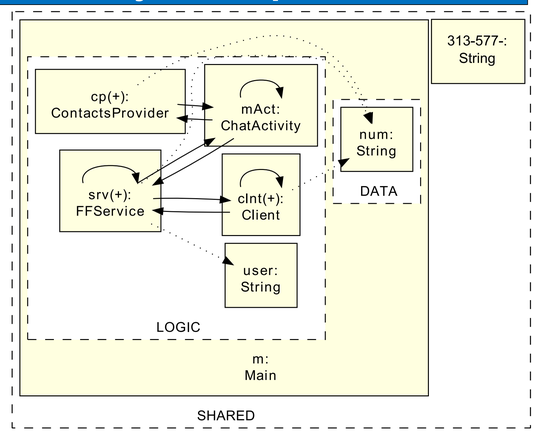
Refine annotation

AChat Code

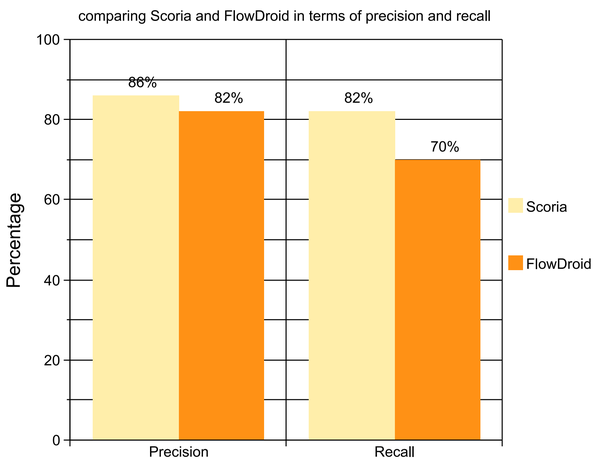
```
class FFService {
  HashMap<String,String> map = ...
  List<String> findByNumber(String num)
{
    List<String> uNames = ...;
    //search for the number in the map
    return uNames;
}
```

```
@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



Results



FlowDroid [Arzt et al., PLDI'2014]

Reasons about information flow at the level of variables. FlowDroid creates an information flow graph: nodes represent variables and edges represent assignments.

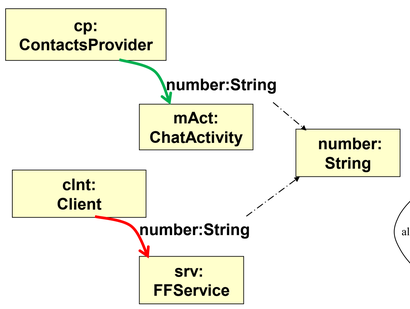
Constraint: FlowDroid looks for transitive information flow from a source to a sink.

(C1,md1, Property1) → (C2,md2, Property2)*

FlowDroid Constraint for AChat :

*(ContactsProvider, getNumbers, **IsConfidential**) ↛* (PrintStream, println, Untrusted)*

Scoria constraint for AChat: the same object that flows from `cp:ContactsProvider` to `mAct:ChatActivity` should not flow from `clnt:Client` to `srv:FFService`.



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