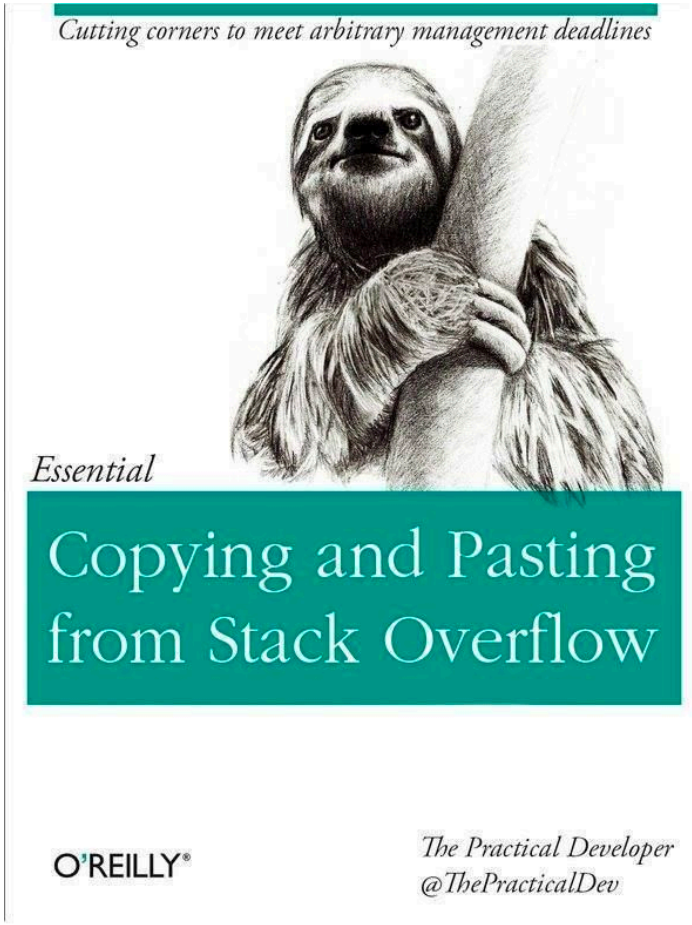
Stack Overflow Considered Harmful?
The Impact of Copy&Paste on Android Application Security

F. Fischer[*], K. Böttinger[*], H.Xiao[*], C. Stransky[†], Y. Acar[†], M. Backes[†], S. Fahl[†]

[*]Fraunhofer AISEC [†]CISPA, Saarland University

Presentation by Kevin Liao

Code copypasta insecure?



Research question

How prolific are security-related code snippets from Stack Overflow in Android applications?

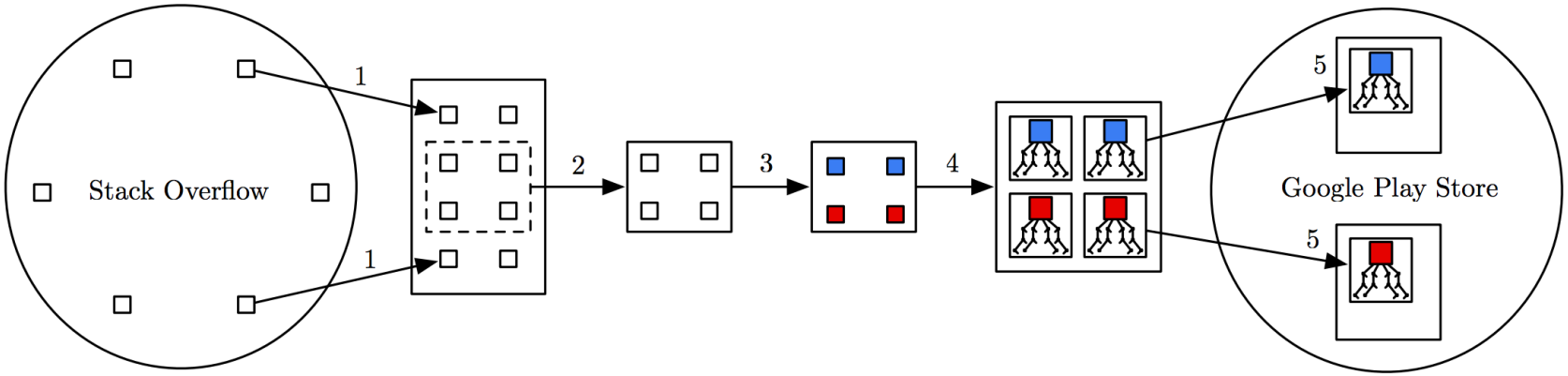
This talk

Rather than discuss results at end…

Present results first, then **analyze the methodology**

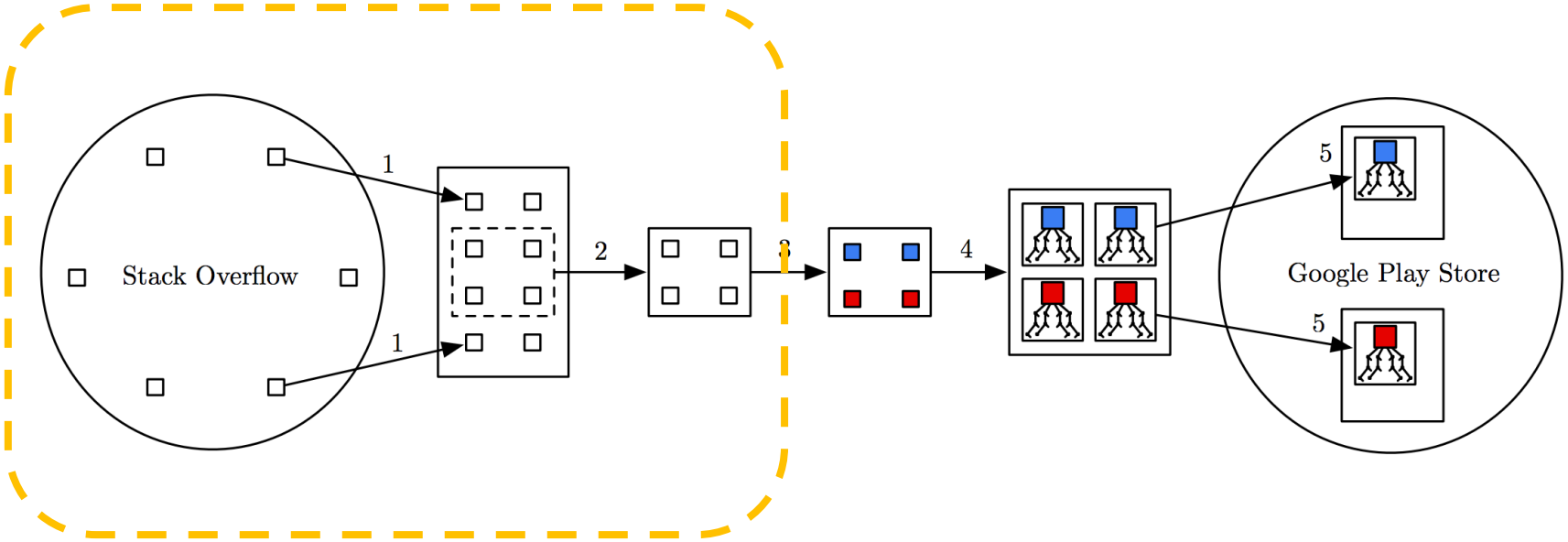
Does the methodology convince us of the results?

The high-level approach

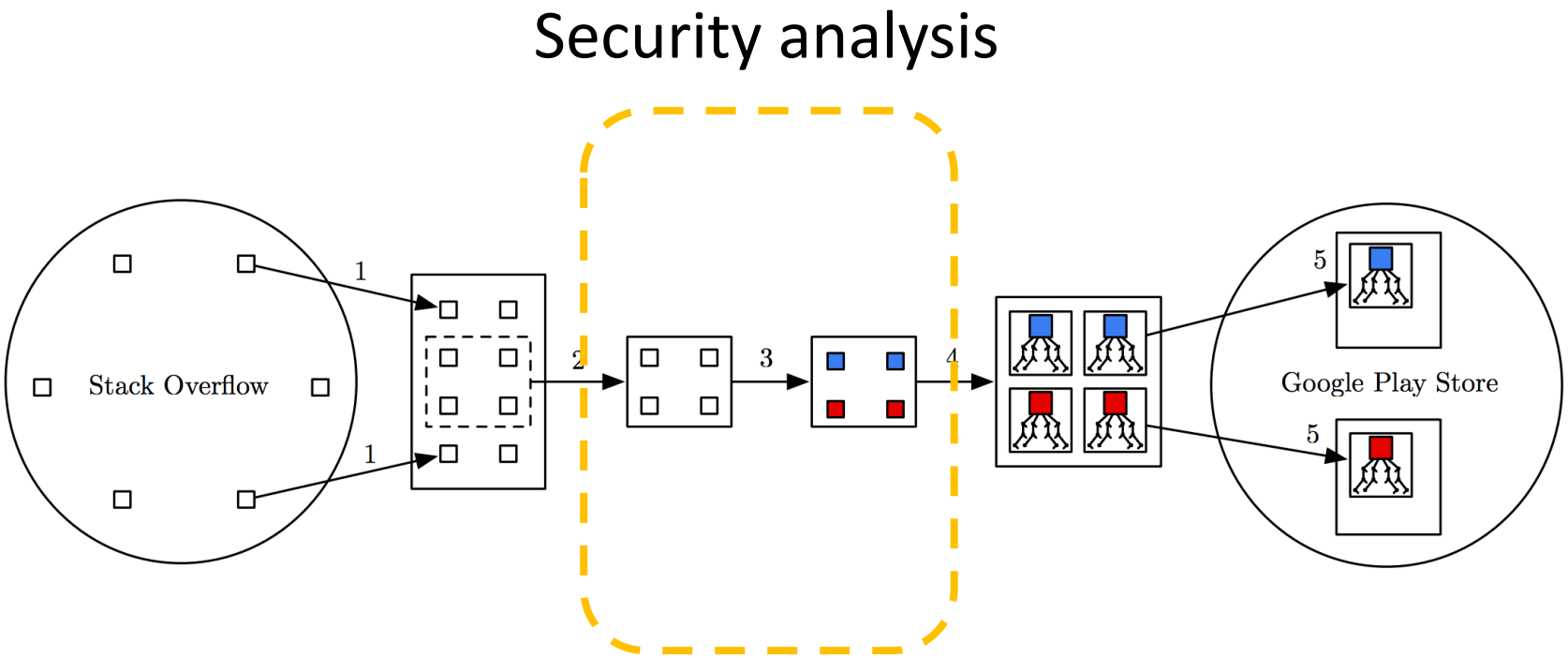


The high-level approach

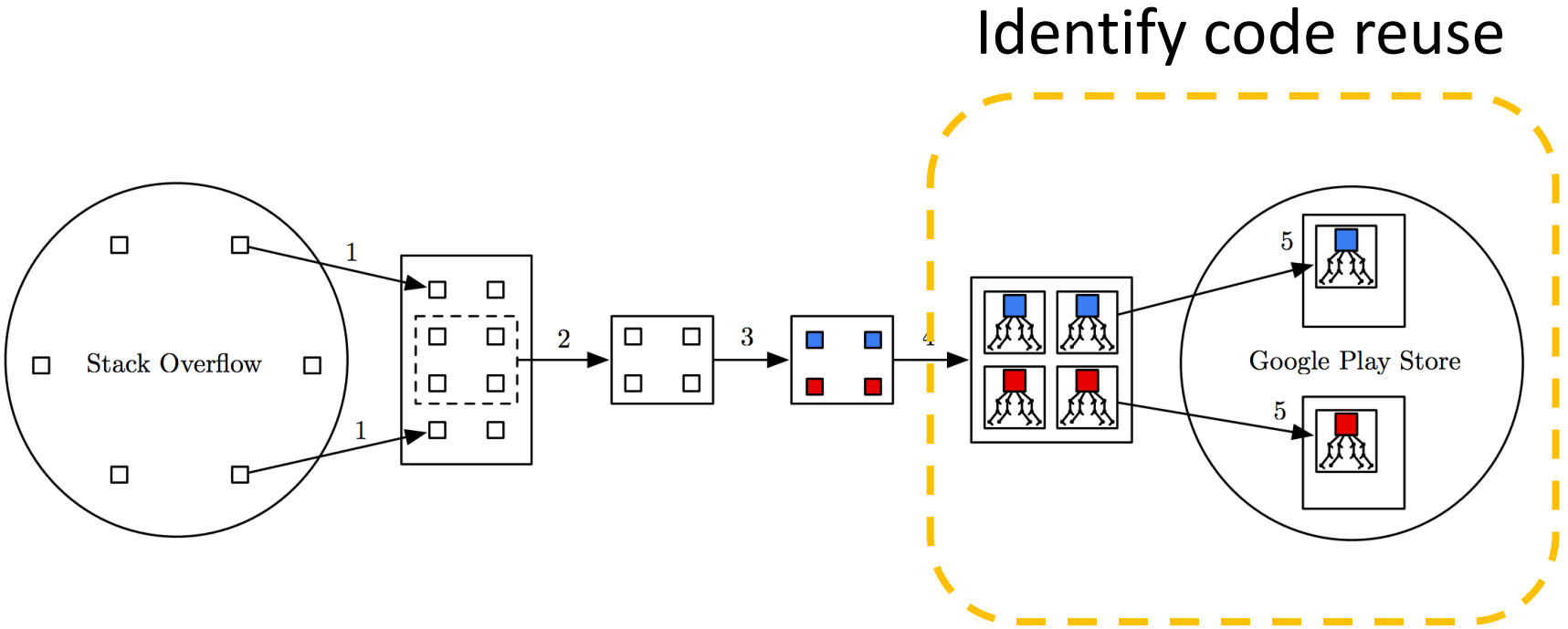
Extract security-related snippets



The high-level approach



The high-level approach



Results: Alarming (potentially)

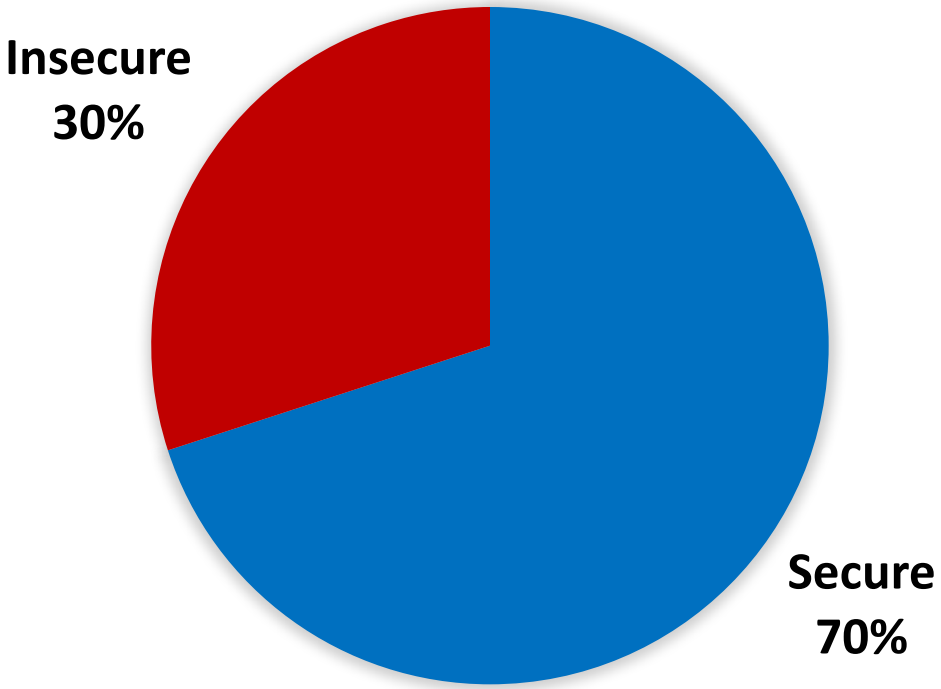
Extracted snippets

30 million posts

2 million Android-related posts

~4,000 security-related snippets

Security classification



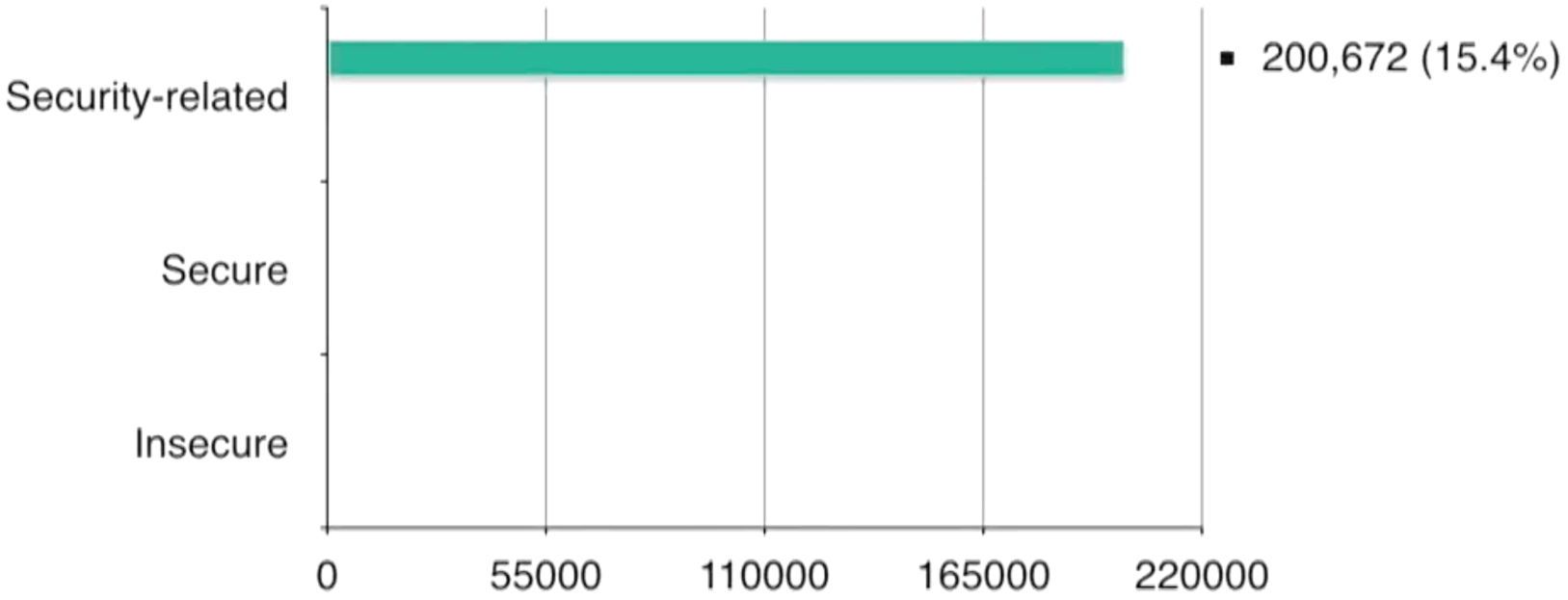
Prevalence of code reuse

Google Play Store

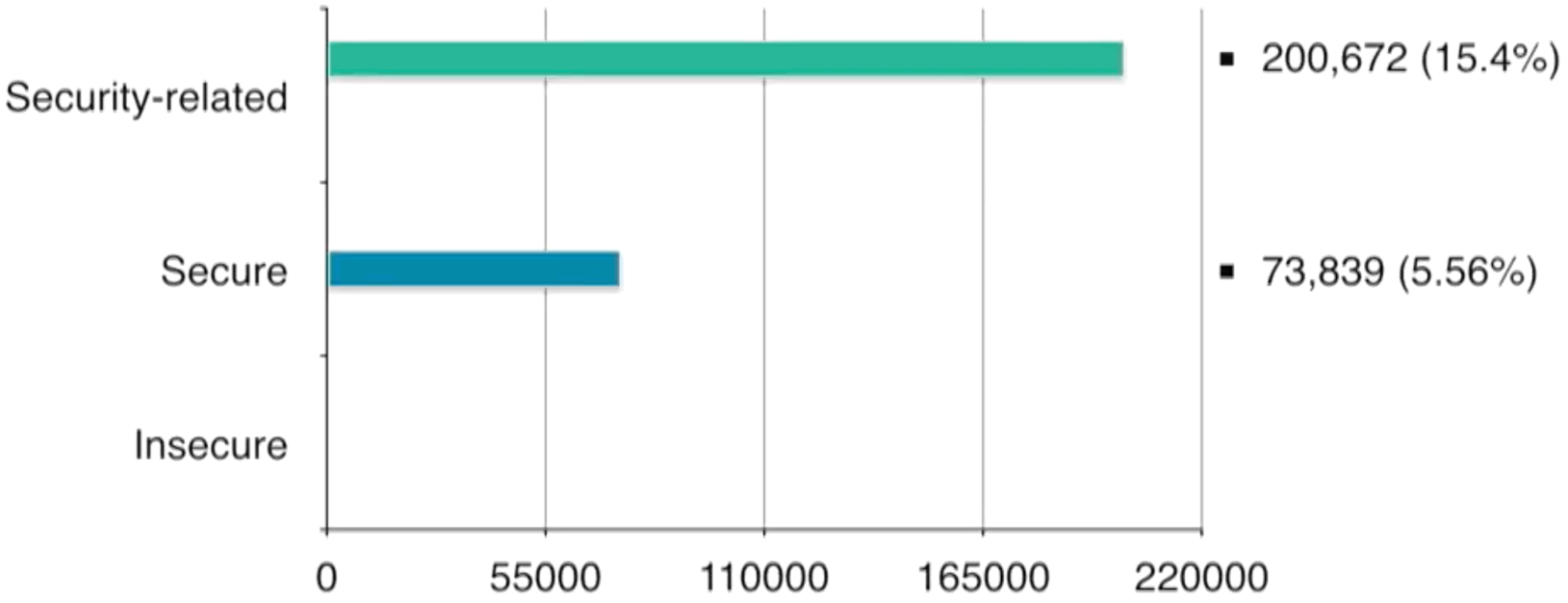
2,673 secure snippets
1,161 insecure snippets

1.3 million free apps

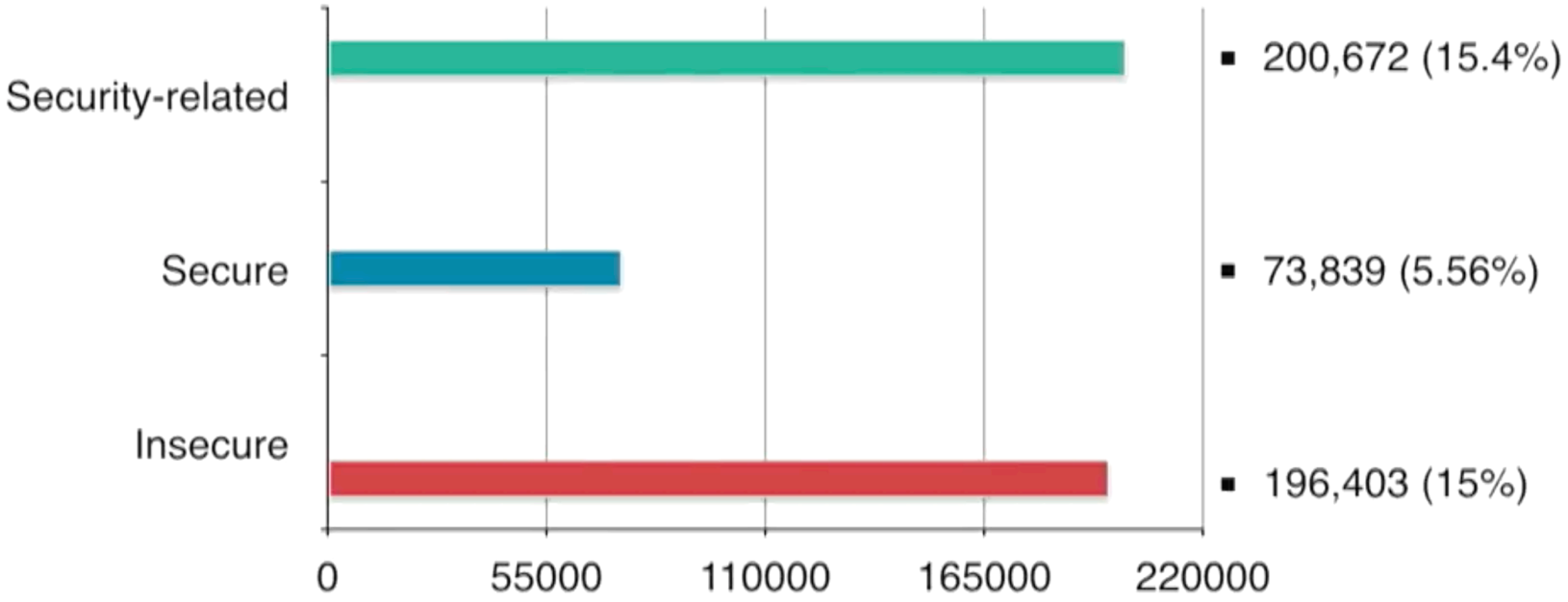
Prevalence of code reuse



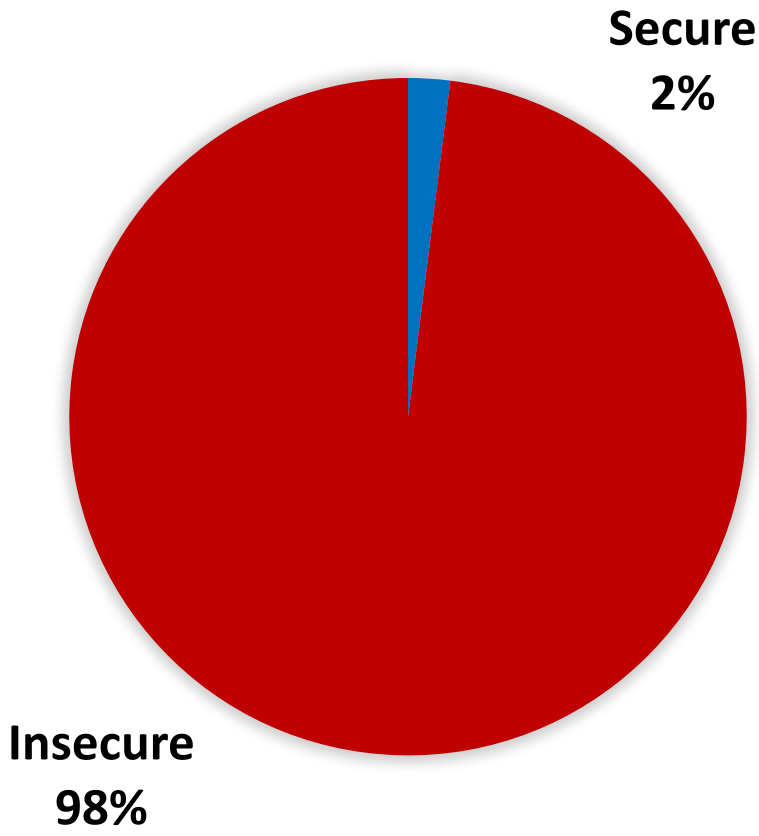
Prevalence of code reuse



Prevalence of code reuse



Apps with security-related snippets



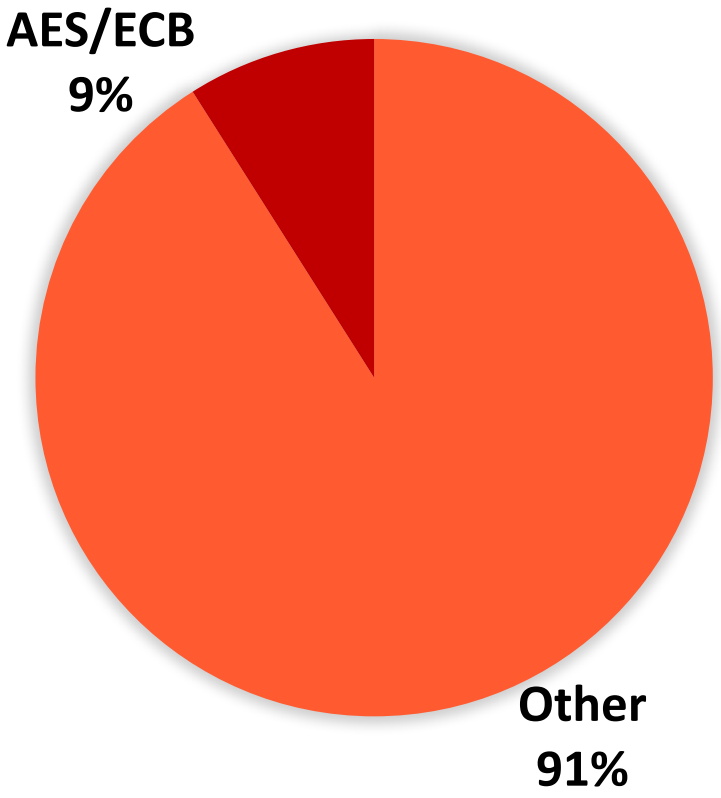
Top-offender? TLS…

- 180k apps w/ empty Trust Manager
- Deactivates server verification
- Can lead to MITM

Next top-offender? Symmetric crypto

Next top-offender? Symmetric crypto

- 18k apps with AES in ECB mode
- Hard-coded keys



Do insecure snippets have lower scores?

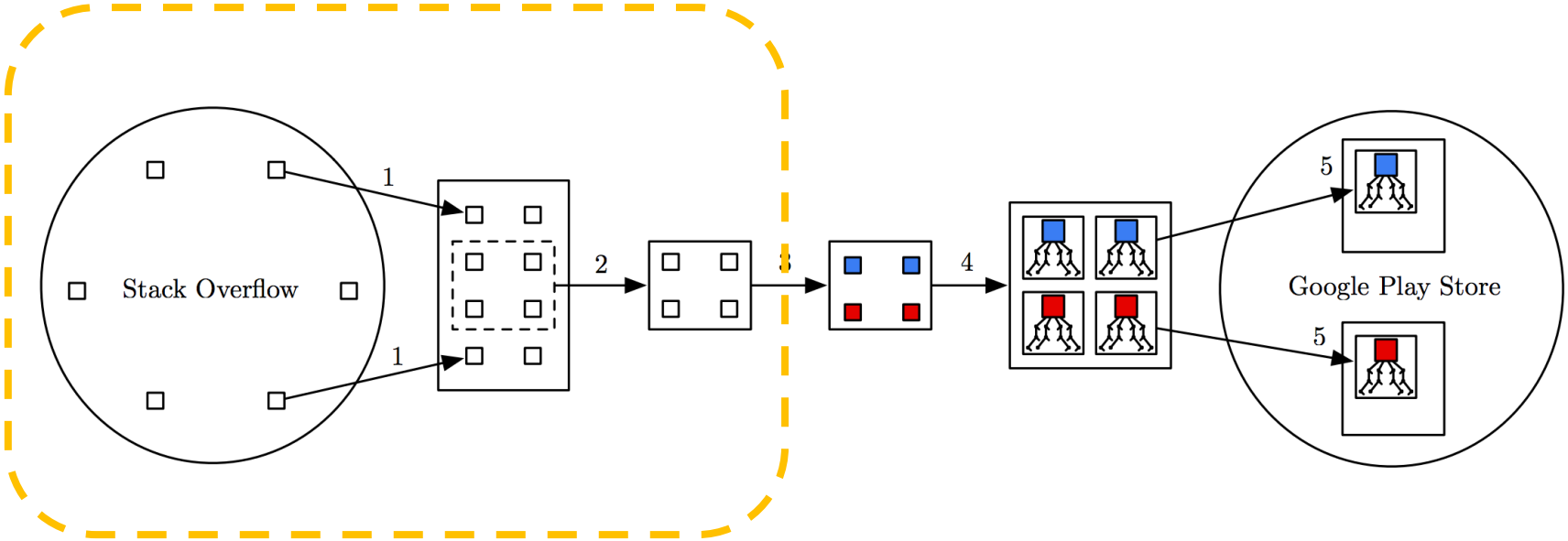
Do insecure snippets **with a warning** have lower scores?

Are high view count/score snippets copy&pasted more?

Are high view count/score snippets **with a warning** copy&pasted **less**?

Discussion of methodology

Extract security-related snippets



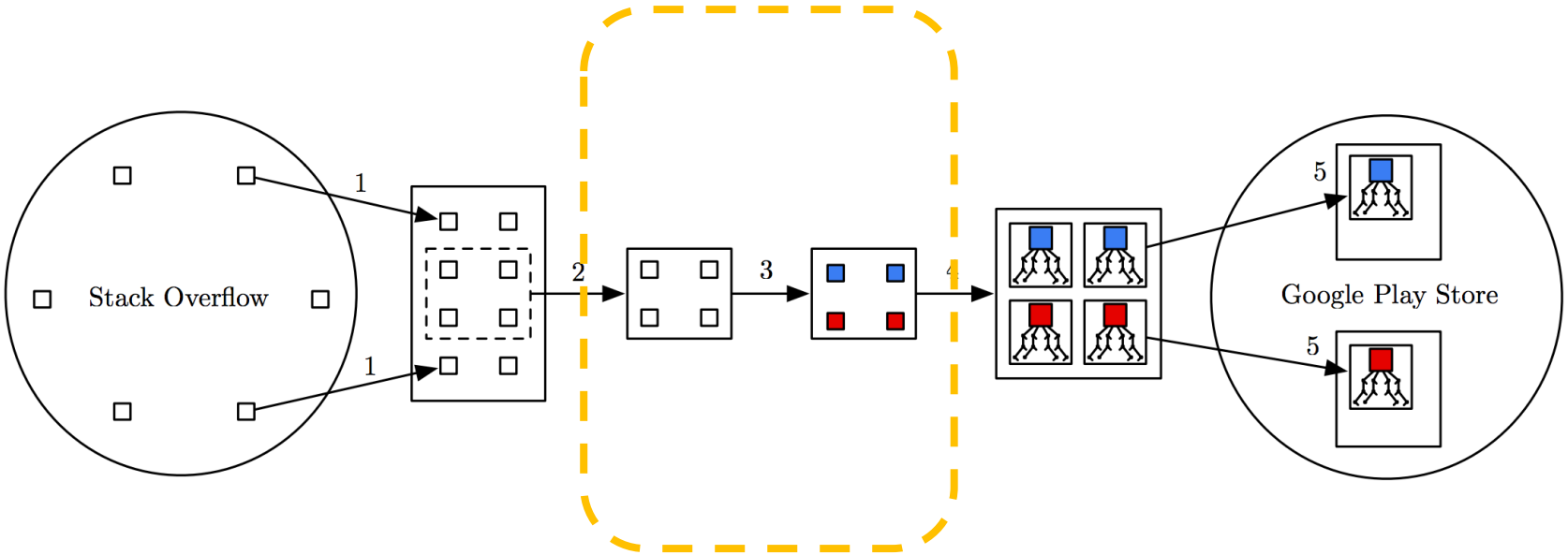
Extract security related-snippets

1. Get all posts with ‘Android’ tag
2. Filter code-snippets that use security APIs
   - TLS/SSL
   - Symmetric/asymmetric crypto
   - RNG
   - Signatures
   - Message digests
   - Authentication/access control

Discuss snippet extraction

Discussion of methodology

Security analysis



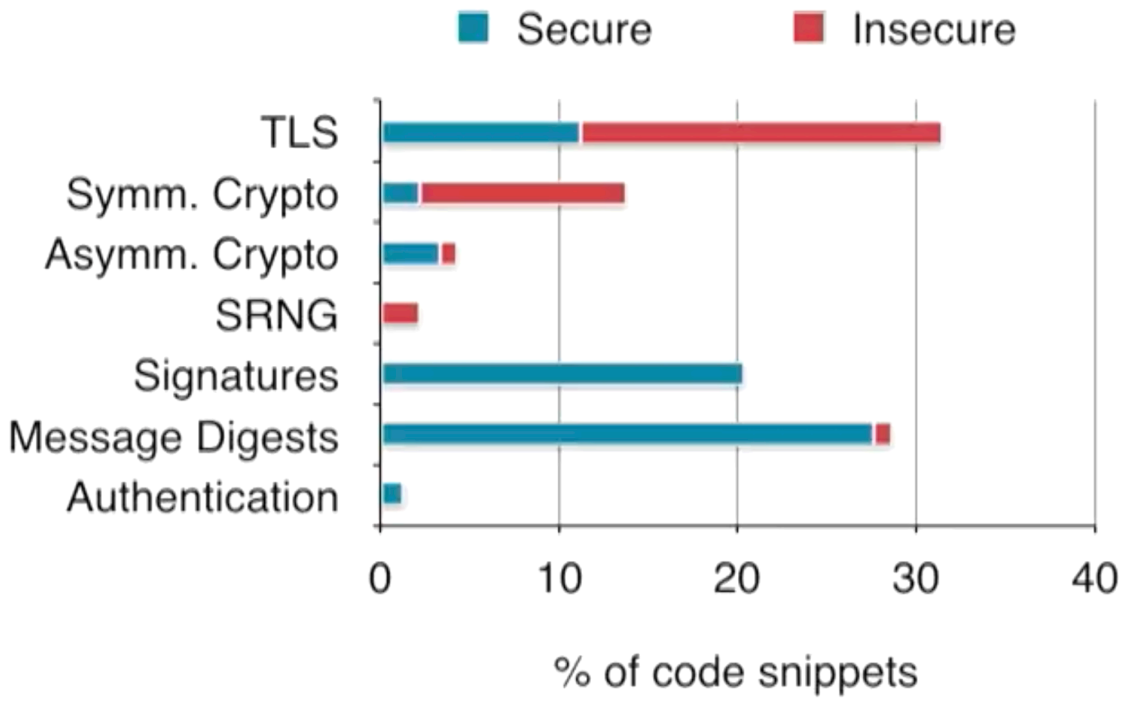
Security analysis

1. Manually label snippets as secure or insecure
2. Train a binary classifier to automatically determine security/insecurity of all snippets

tl;dr for labeling rules

- SSL/TLS: Use TLS v1.1 or greater; don’t use old crypto
- Symmetric: Don’t use old crypto; don’t use ECB; don’t use static/zeroed/derived keys or IVs
- Asymmetric: Use >=2048 bit RSA; use >= 244 bit ECC
- Hashing: Don’t use MD-family
- RNG: Use crypto-secure RNG; securely random seed

Security score of training set



Train SVM binary classifier

Feature selection

- Based on tf-idf
- “The features rely merely on the vocabulary level of input code snippets, without even understanding how they are functioning.”
- Claim: Can be more accurate and more scalable than rule-based methods

TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Term frequency: Number of times term t appears in a doc, d

Inverse document frequency

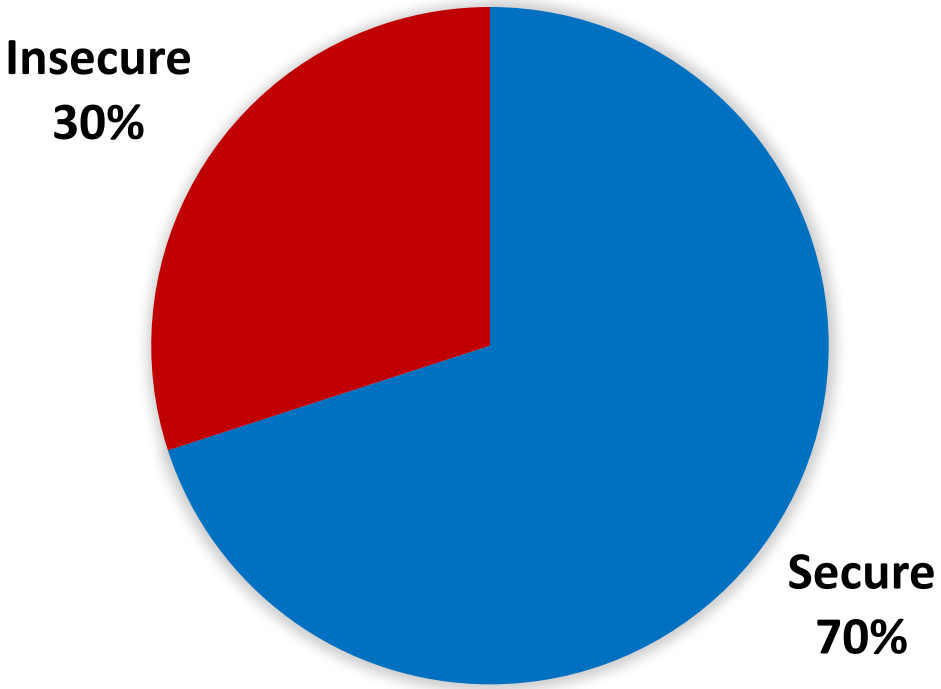
$$\log \frac{1 + n}{1 + \text{df}(d, t)} + 1$$

n ← # of documents

df(d, t) ↖ Document frequency of the term t

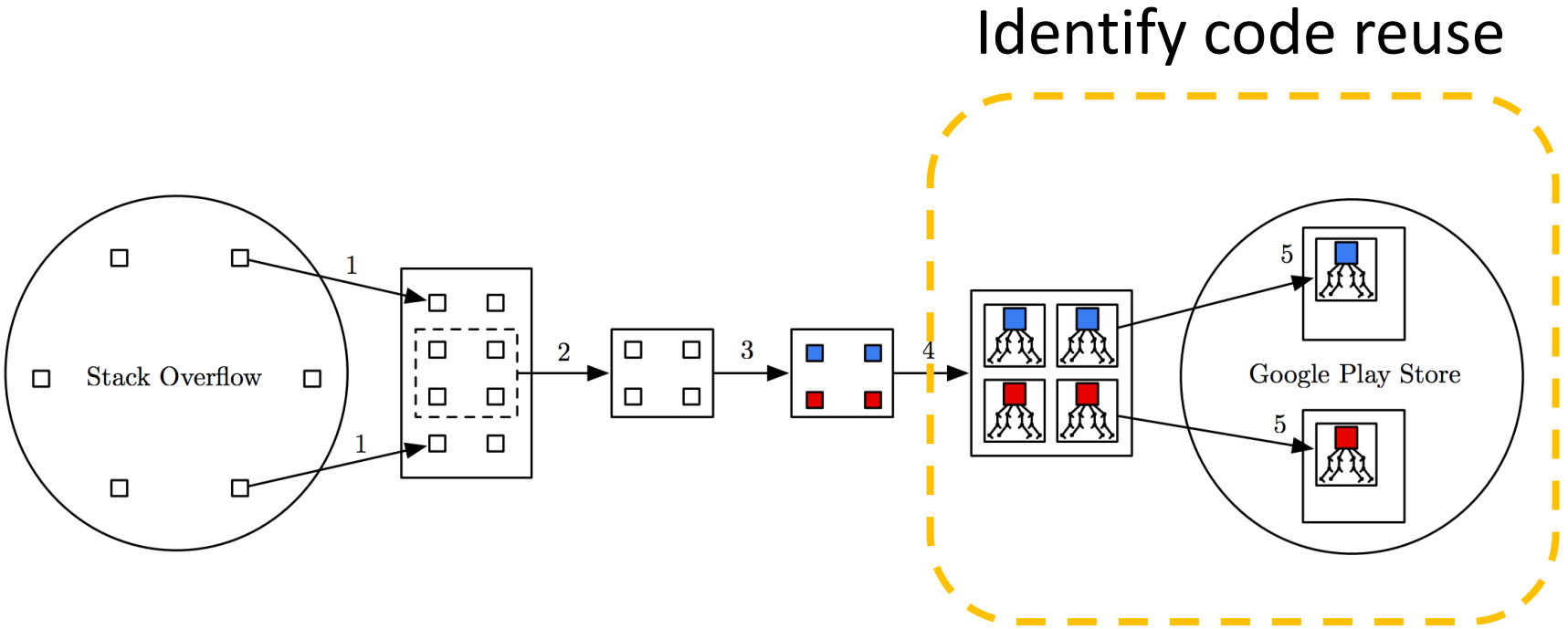
https://chrisalbon.com/machine_learning/preprocessing_text/tf-idf/

Security classification



Discuss security classification

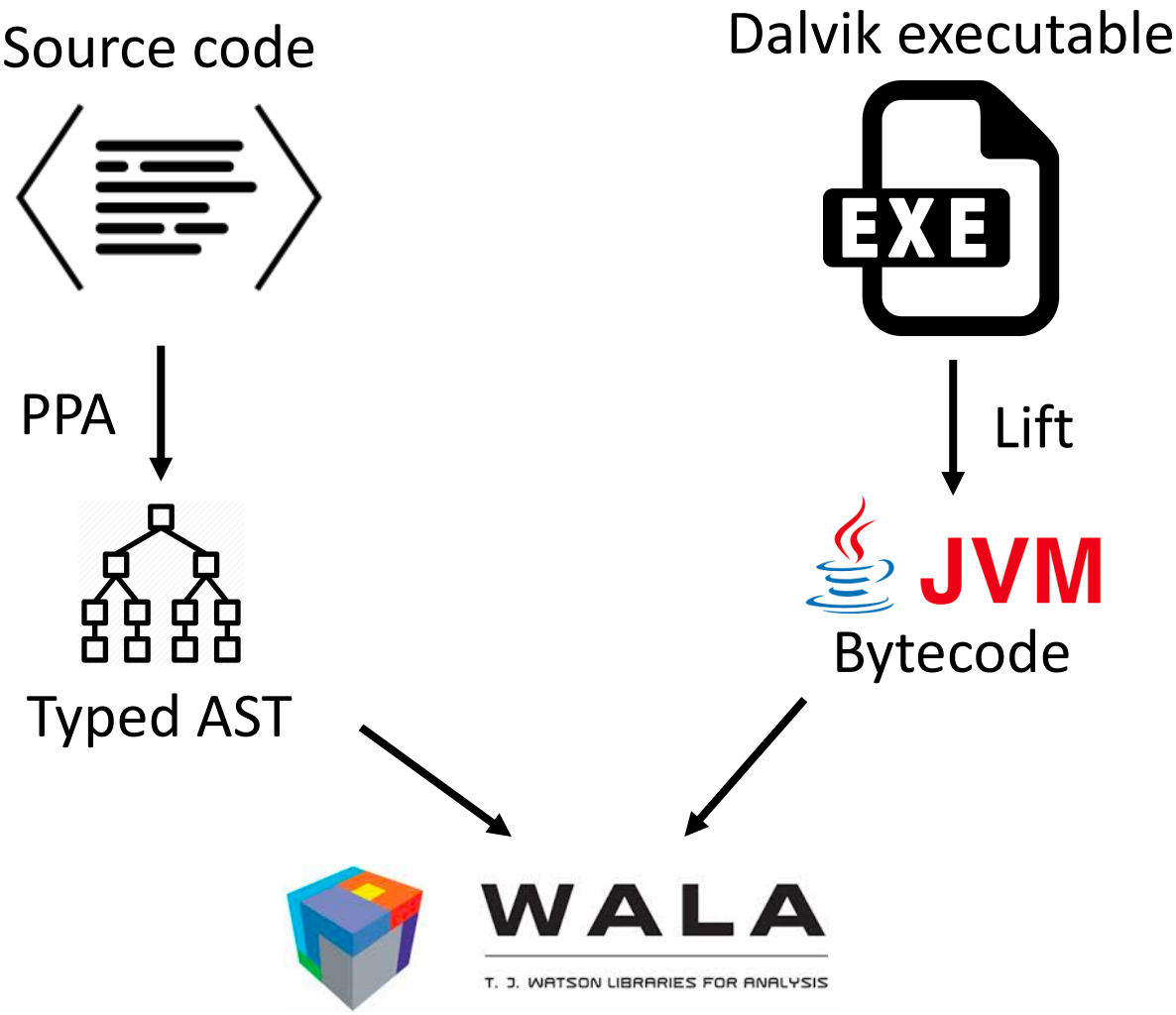
Discussion of methodology



Identify code reuse

1. Transform source code and Dalvik executables into same IR
2. Identify similar code snippets using Program Dependency Graphs (PDGs)

IR transformation



Program Dependency Graphs

- Generate PDG for each method
- Nodes: Statements in methods
- Edges: Data and control dependence

Dependency edges

Data: S2 depends on S1, since A read in S2.

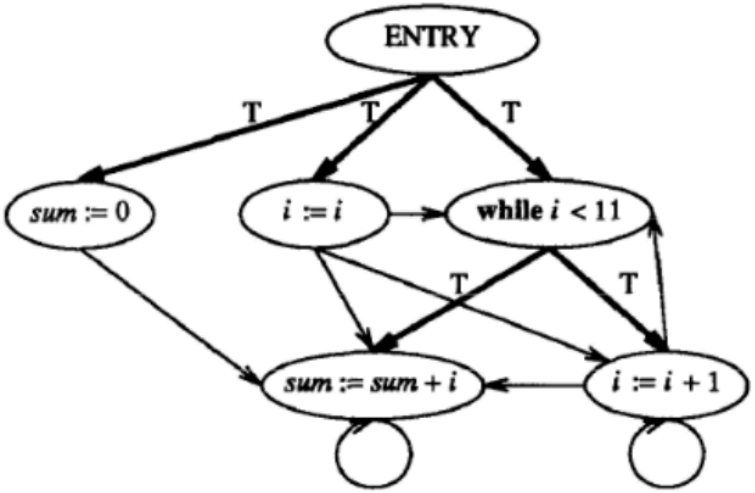
```
S1: A = B * C
S2: D = A * E + 1
```

Control: S2 depends on A, since A determines S2's execution.

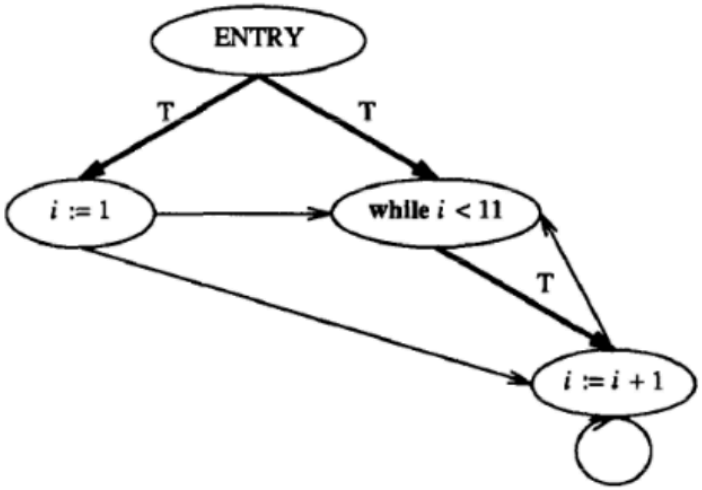
```
S1: if (A) then
S2:     B = C * D
    endif
```

Examples of PDGs

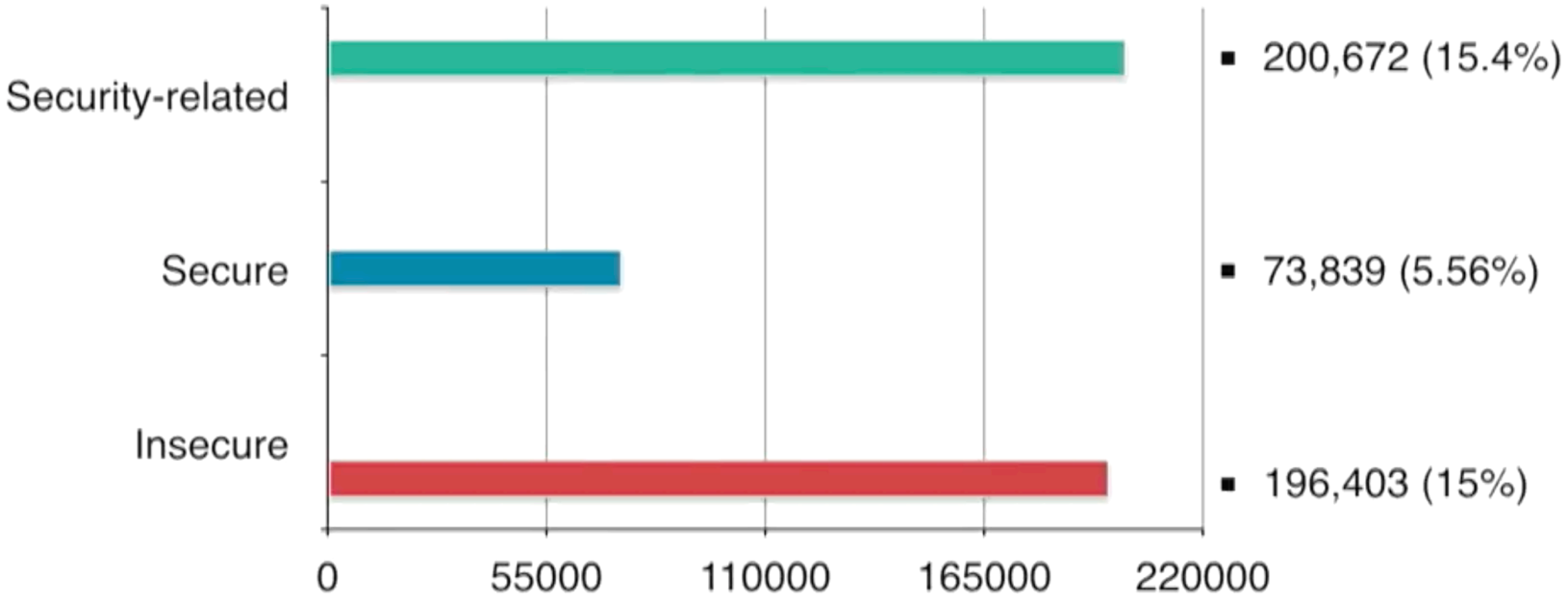
```
program Main
    sum := 0;
    i := 1;
    while i < 11 do
        sum := sum + i;
        i := i + 1
    od
end
```




```
program Main
   i := 1;
   while i < 11 do
      i := i + 1
   od
end
```



Prevalence of code reuse



Discuss identification of code reuse

Final discussion

- About results?
- About methodology?
- About future work?