Security Audit for Contact Tracing Apps applied to the Corona-Warn-App

WAW SE, 10.09.2020

DLR

Contact Tracing Apps

Goal: Smartphones should help to track contacts of Covid-19 infected persons.

- Current approach:
  - Using pen, paper and telephone to trace contacts
    - extremely time consuming
    - extremely personnel-intensive
    - contact information often incomplete and inaccurate

Central vs. Decentral -- Where and who is irrelevant, what matters is how close and how long

Pan European Privacy Preserving Proximity Tracing (Pepp-PT)

- A **central** server - an all-knowing authority -- stores all contacts
- Based on Bluetooth Low Energy
- **Goal**: One technology as a basis, many national apps.

-- A centralized approach may invite to misuse

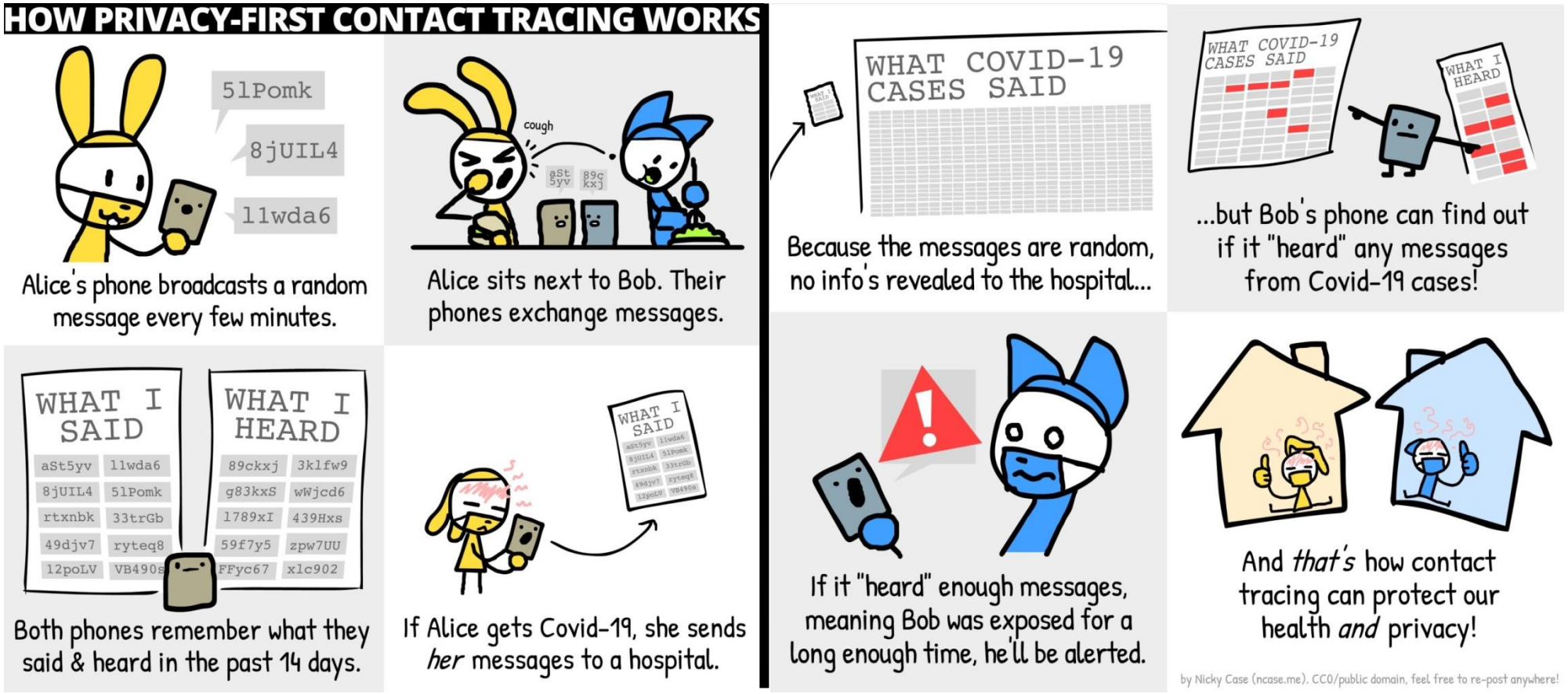
-- With this information it would be possible to create a social graph.

+ Changes to the infection risk calculation can be quickly implemented on a central server

Decentralized Privacy-Preserving Proximity Tracing (DP-3T)

- Open Source - https://github.com/DP-3T
- Decentralized system -- personal data and calculations are stored on the phone
- Based on Bluetooth Low Energy

-- All users get the IDs of the infected persons - are temporary IDs personal data?



https://github.com/DP-3T/documents/tree/master/public_engagement/cartoon

Exposure Notification Framework

- Developed by Apple and Google
- The API is available exclusively to registered government health authorities (one app per country)
- Enables the sending and receiving of temporary IDs via BLE
- Elimination of technical problems:
  - both systems were not designed to perform continuous Bluetooth scans
  - ensuring the interoperability of iOS and Android
- Highest priority: privacy, control for the user & battery-saving
  - records no GPS data
  - user must explicitly agree to the terms of use and can deactivate the function at any time
- Removing the interface as soon as it is no longer needed

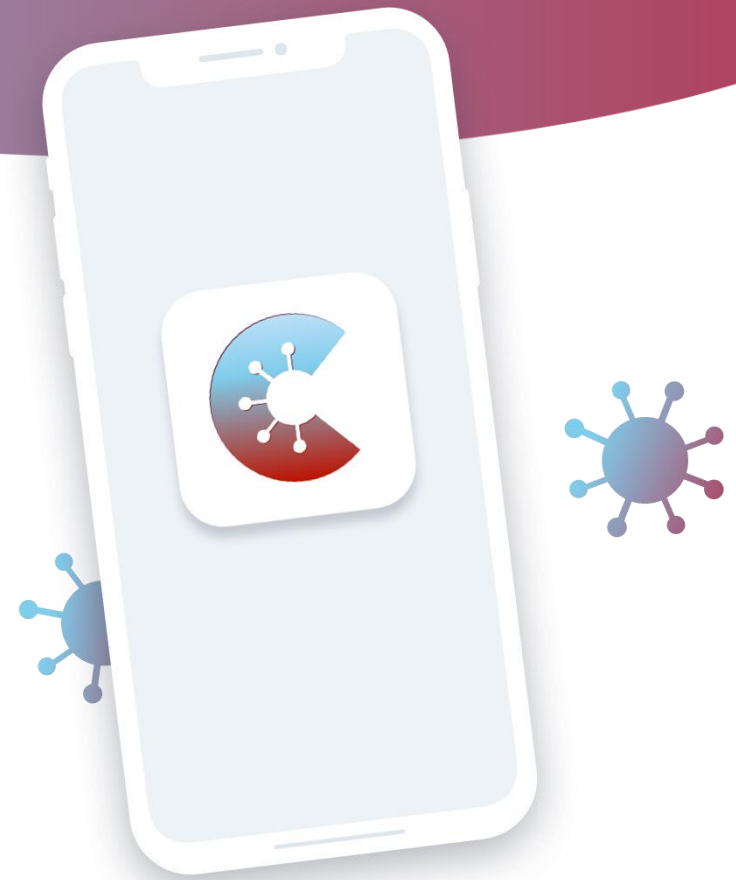
Exposure Notification Framework - Configuration Parameters

Formula used to calculate the Exposure Risk Value - **totalRiskScore**

- Parameters to calculate a risk for each exposure incident (adjustable by health authorities):
  - **Transmission Risk**: reflect the status of infection in the affected user and its effect on risk of transmission; defined in the smartphone app
  - **Duration**: the duration of the Bluetooth contact
  - **Days**: the time that has passed since the patient's mobile phone had its last Bluetooth contact with the mobile phone of the person to be warned
  - **Attenuation**: the strength of the Bluetooth signal (in dBm)

- Corona-Warn-App: 80% correct and 20% incorrect reports

Corona-Warn-App



- Germany's official *Exposure Notification App* (www.coronawarn.app)
- Developed by SAP and Deutsche Telekom

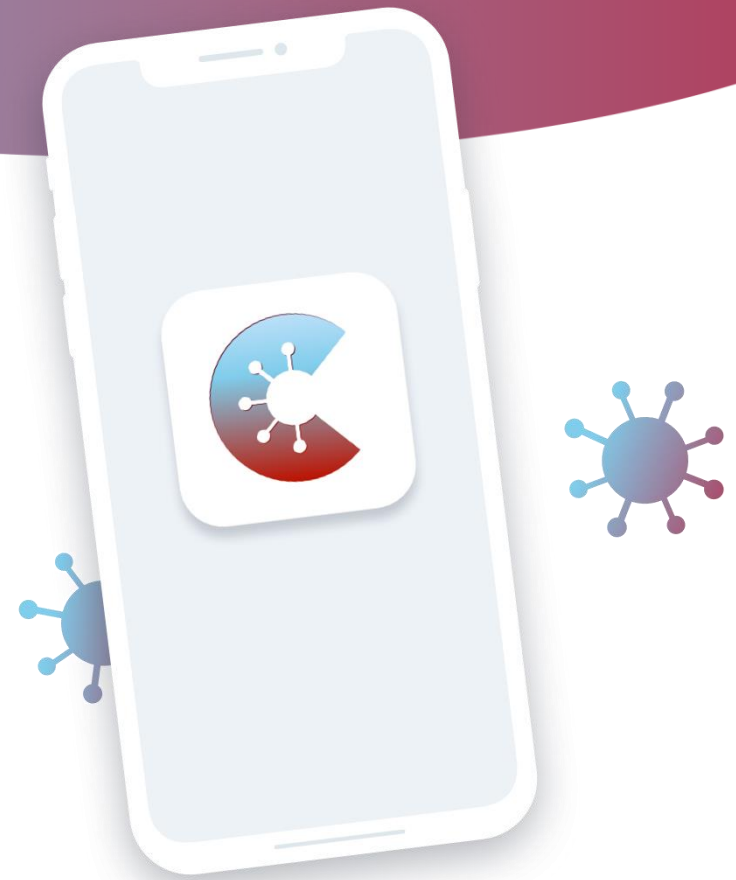
- Open source software on Github (https://github.com/corona-warn-app)
- Apache 2.0 license – Standard SAP Open Source Lizenz
- 17.8 Mio downloads (01.09.2020)

- 20 € Mio development costs
  - 2,5 – 3,5 € Mio per month for hosting and multilingual telephone hotline

Corona-Warn-App

„Ich war froh, dass wir mit der App nichts zu tun hatten, weil wir dann auch nichts falsch machen können.“

- SAP-Employee -



Two Companies - One Mission: SAP & Deutsche Telekom

- Processes are much more efficient
  “Normalerweise dauert das schon noch etwas länger.” (SAP)

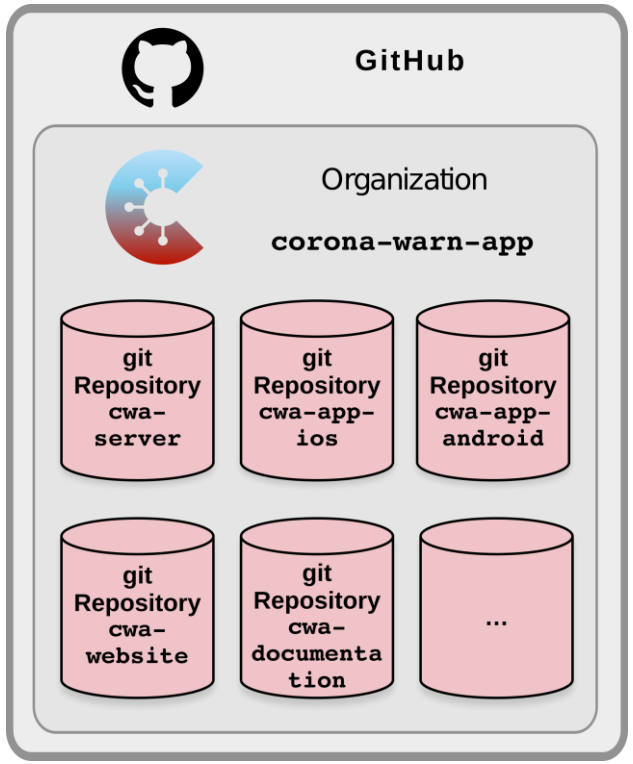
- Benefit Home Office -- „alle im selben Internetbüro“ (SAP)
  - no noticable company or team boundaries

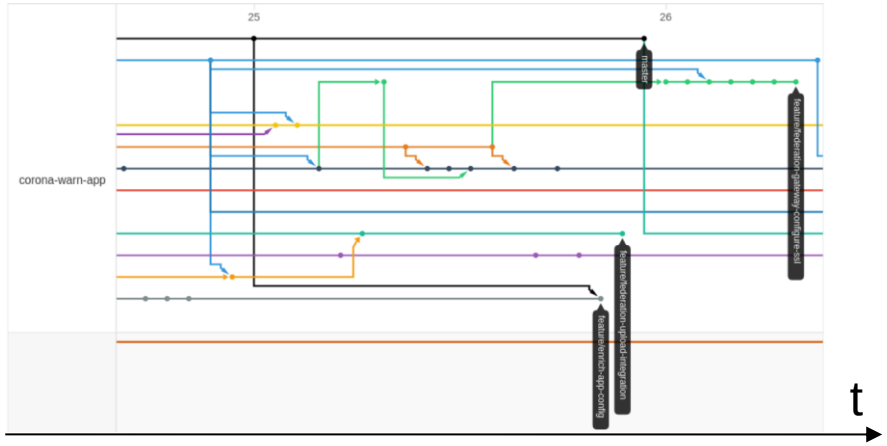
- Open Source Culture:
  - SAP already has experience with open source development (development & contributor)
  - SAP Open Source Office provided a “Rahmenwerk” (Issuetemplates, Code of Conduct, etc.)
  - all repositories public on GitHub

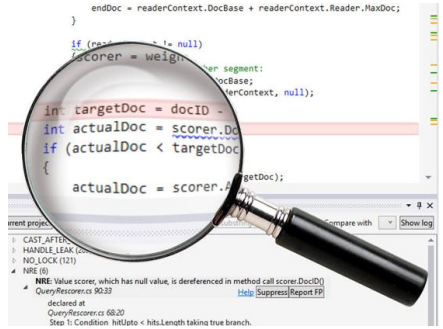
Security Audit for Contact Tracing Apps

The Idea

- Use information from the software development process (repositories)
- and analyse the code at different stages to find possible security weak points



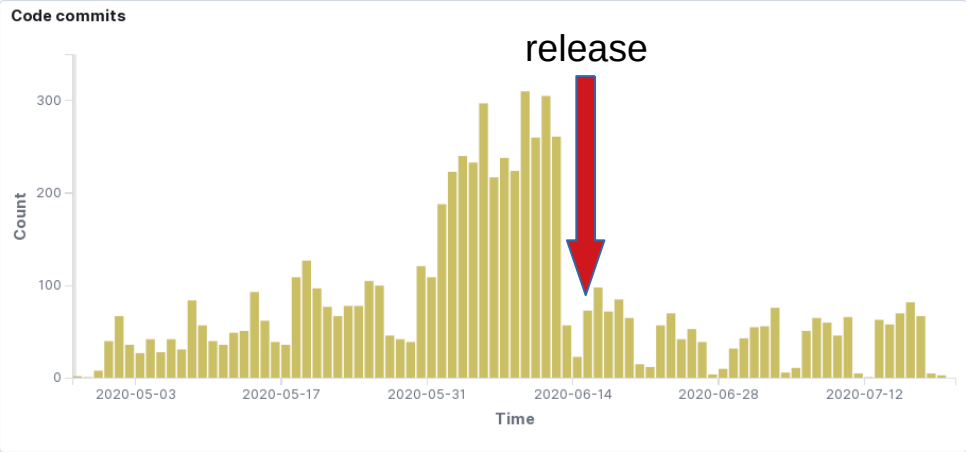




Static Analysis

- Checking program code for errors or bugs without executing it
  - Not testing!
- Different types:
  - Linter
    - Formatting, Code complexity, ...
  - Coding rules
    - Bug detection, code metrics, code duplicates, performance checks
  - Formal verification
  - Taint analysis
    - Perform a data-flow analysis

Security Audit of the Corona-Warn-App
Example



Code commits of all cwa-repositories

```
115         Timber.d("Diagnosis Keys will be submitted.")
116         withContext(Dispatchers.IO) {
117             Timber.d("Writing ${keysToReport.size} Keys to the Submission Payload.")
118             val submissionPayload = KeyExportFormat.SubmissionPayload.newBuilder()
119                 .addAllKeys(keysToReport)
120                 .build()
121             var fakeHeader = "0"
122             if (false) fakeHeader = Math.random().toInt().toString()
```

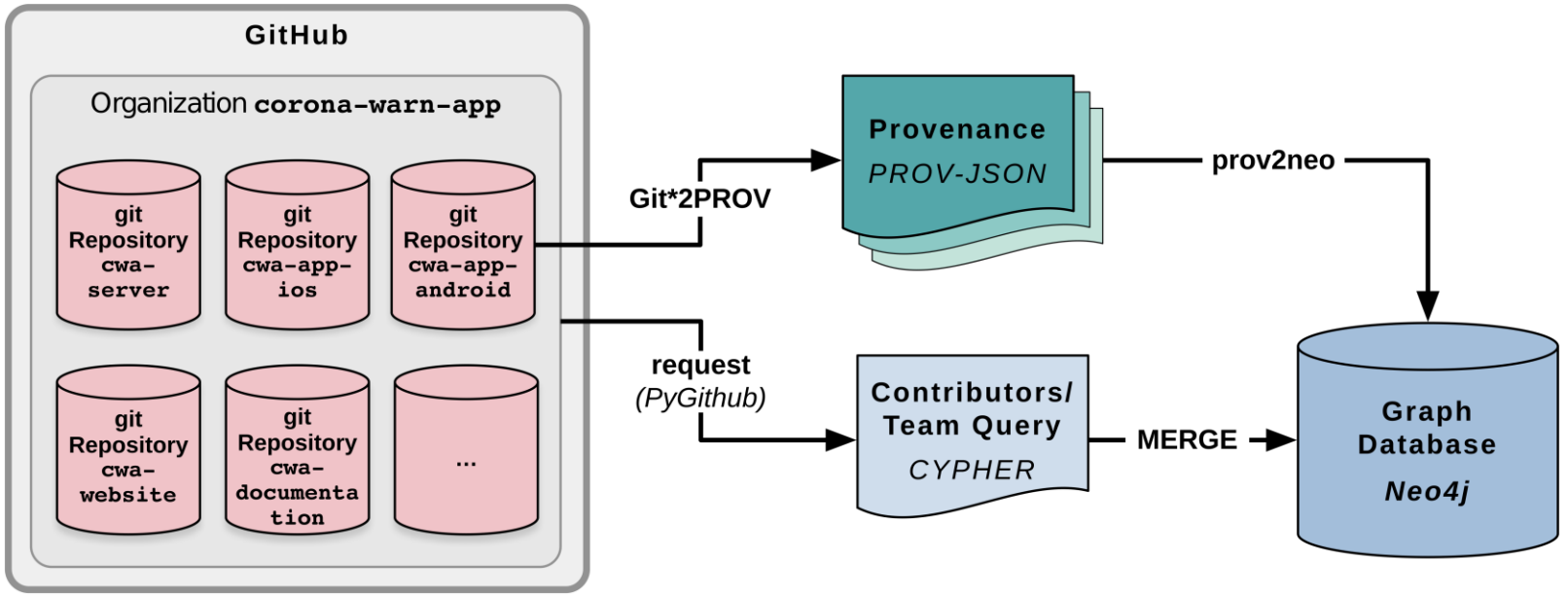
Remove this useless "if" statement. Why is this an issue? 4 days ago L122

Bug Major Open Not assigned 2min effort No tags

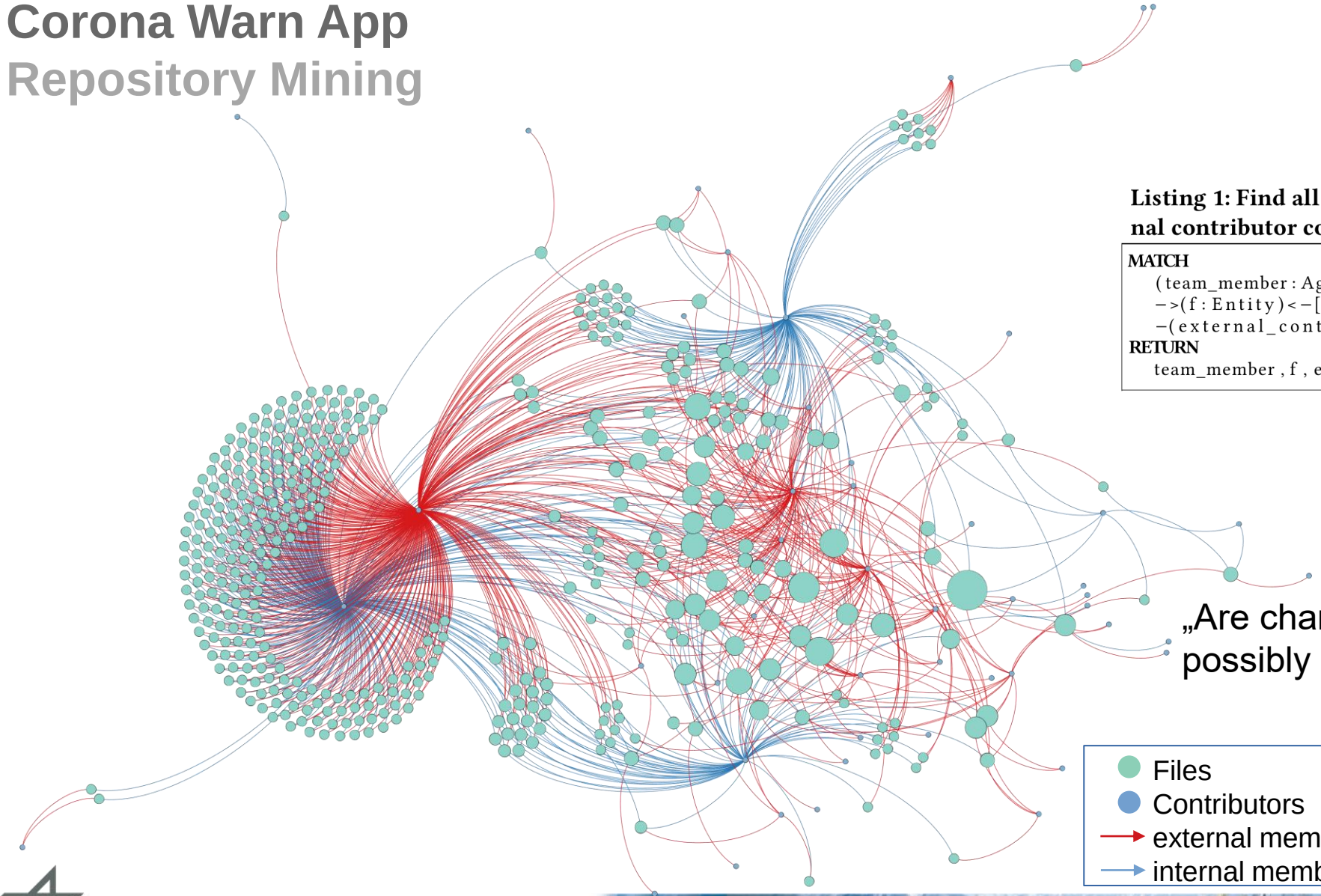
Introduced bug
‘CWE-561 – Dead code’,
‘CWE-570 – expression is always false’
detected by SonarQube Scanner.

Security Audit of the Corona-Warn-App
Repository Mining

- SC: Provide provenance information as indicator for possible analysis points



Corona Warn App
Repository Mining



Listing 1: Find all files where a team member AND an external contributor contributed changes.

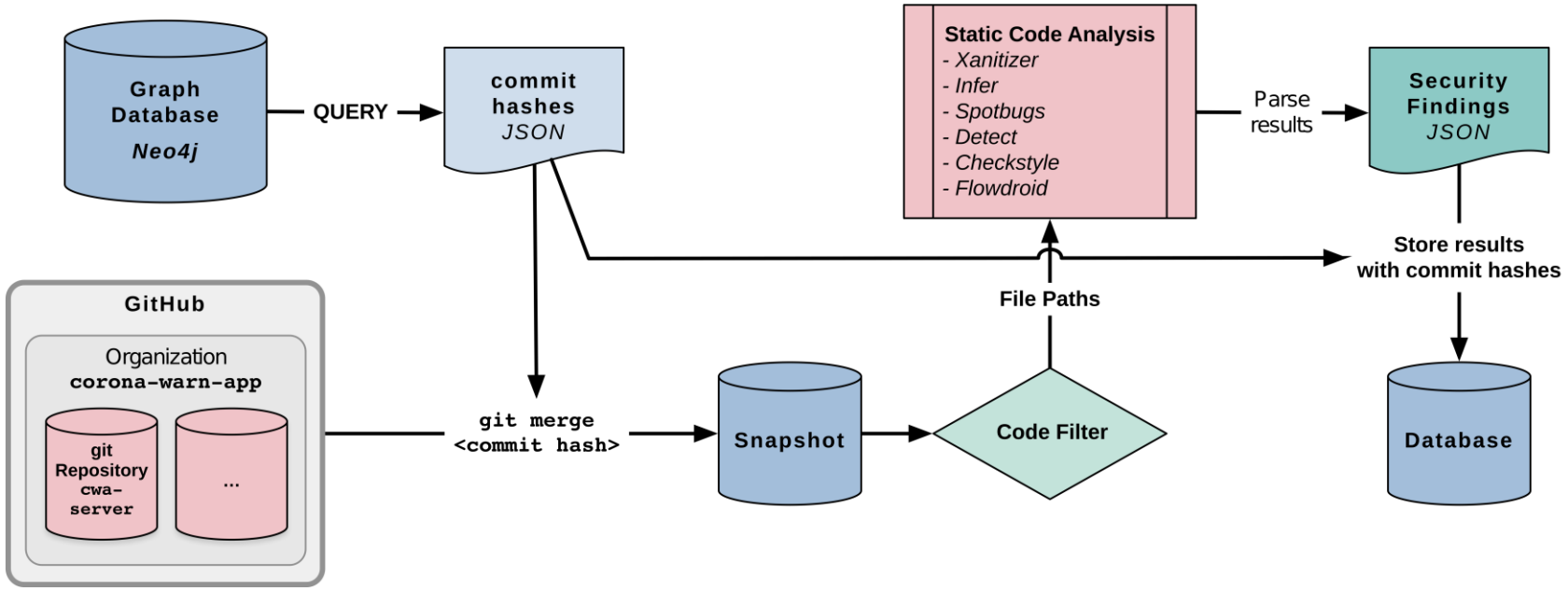
```
MATCH
  (team_member:Agent)-[r1:CONTRIBUTES_TO {role: 'team'}]
  ->(f:Entity)<-[r2:CONTRIBUTES_TO {role: 'contributor'}]
  -(external_contributor:Agent)
RETURN
  team_member, f, external_contributor
```

„Are changes from external contributors possibly more insecure?“

- Files
- Contributors
- external members
- internal members

Corona Warn App
Code Analysis

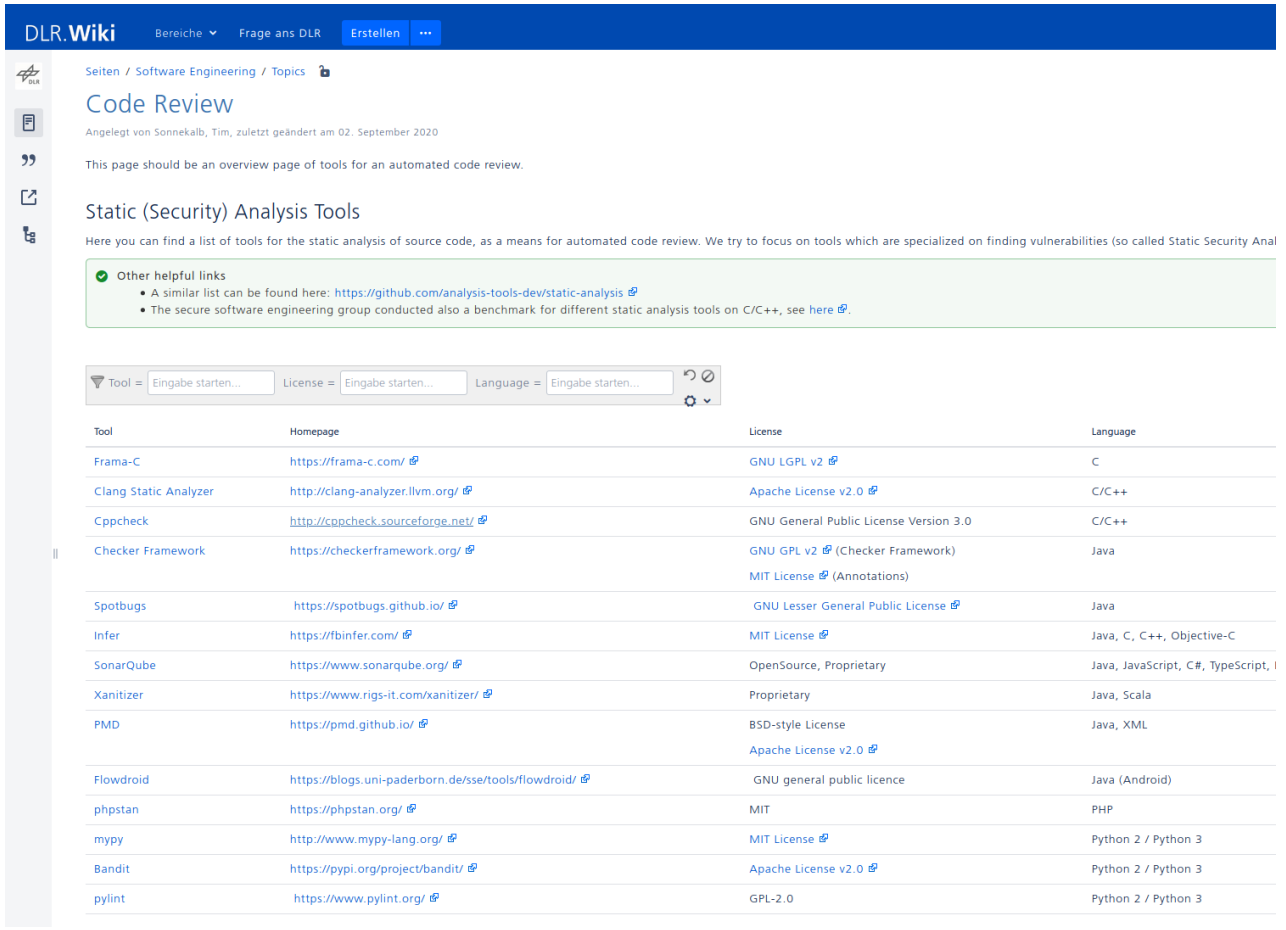
- DW: Analysis controlled by commits as interface to provenance analysis



- Future work:
  - Automation of the code analysis pipeline
  - Addition of code change metrics from *code diff* graphs

SE Wiki
Code Review Tools

- We created an overview page with code review tools in the SE wiki
- One component of a Secure Software Development Lifecycle
- We plan to provide more information about Secure Programming in the Wiki



DLR.Wiki Bereiche Frage ans DLR Erstellen

Seiten / Software Engineering / Topics

Code Review

Angelegt von Sonnekalb, Tim, zuletzt geändert am 02. September 2020

This page should be an overview page of tools for an automated code review.

Static (Security) Analysis Tools

Here you can find a list of tools for the static analysis of source code, as a means for automated code review. We try to focus on tools which are specialized on finding vulnerabilities (so called Static Security Anal

Other helpful links
- A similar list can be found here: https://github.com/analysis-tools-dev/static-analysis
- The secure software engineering group conducted also a benchmark for different static analysis tools on C/C++, see here.

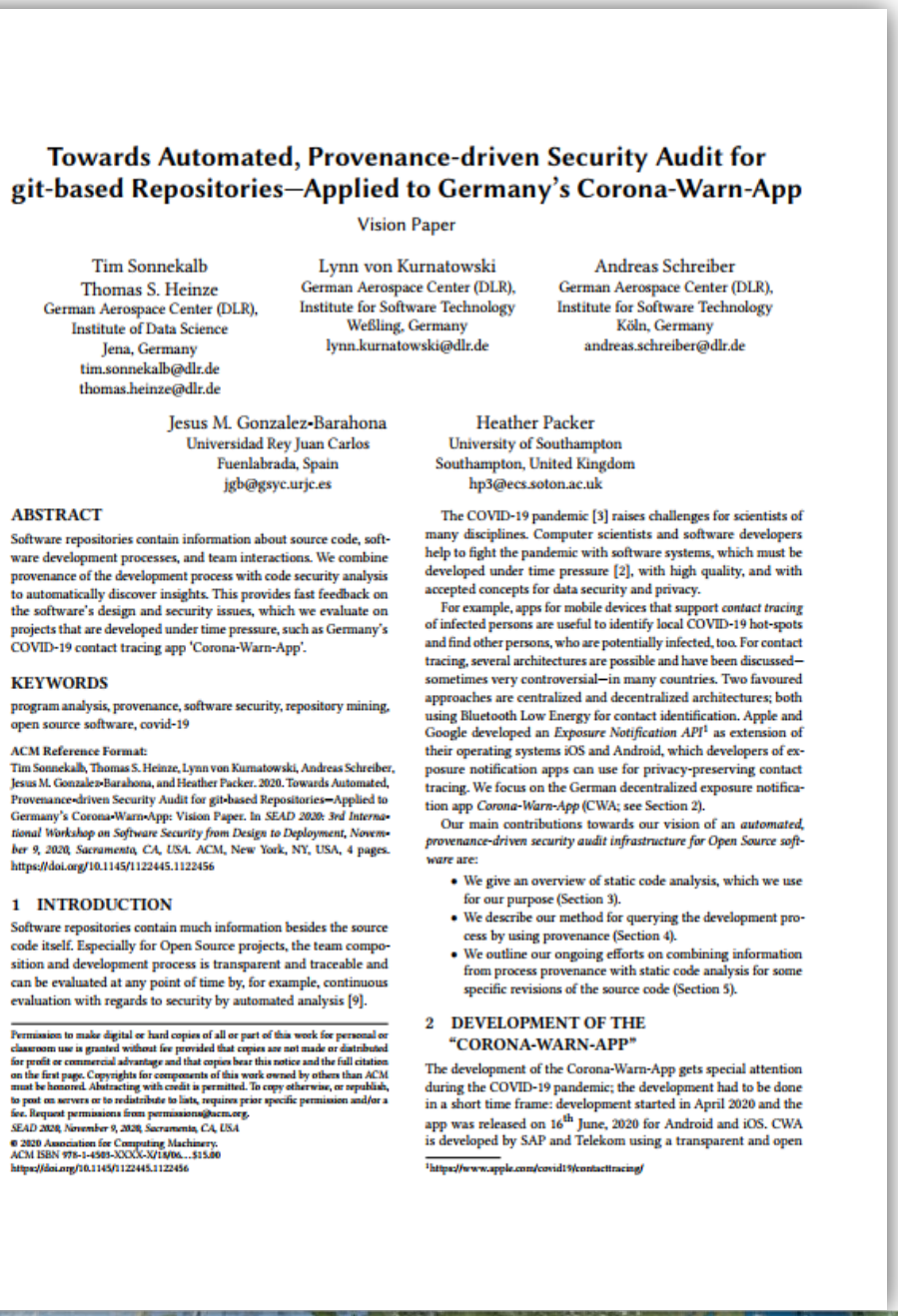
Tool = Eingabe starten... License = Eingabe starten... Language = Eingabe starten...

Tool	Homepage	License	Language
Frama-C	https://frama-c.com/	GNU LGPL v2	C
Clang Static Analyzer	http://clang-analyzer.llvm.org/	Apache License v2.0	C/C++
Cppcheck	http://cppcheck.sourceforge.net/	GNU General Public License Version 3.0	C/C++
Checker Framework	https://checkerframework.org/	GNU GPL v2 (Checker Framework) MIT License (Annotations)	Java
Spotbugs	https://spotbugs.github.io/	GNU Lesser General Public License	Java
Infer	https://fbinfer.com/	MIT License	Java, C, C++, Objective-C
SonarQube	https://www.sonarqube.org/	OpenSource, Proprietary	Java, JavaScript, C#, TypeScript,
Xanitizer	https://www.rigs-it.com/xanitizer/	Proprietary	Java, Scala
PMD	https://pmd.github.io/	BSD-style License Apache License v2.0	Java, XML
Flowdroid	https://blogs.uni-paderborn.de/sse/tools/flowdroid/	GNU general public licence	Java (Android)
phpstan	https://phpstan.org/	MIT	PHP
mypy	http://www.mypy-lang.org/	MIT License	Python 2 / Python 3
Bandit	https://pypi.org/project/bandit/	Apache License v2.0	Python 2 / Python 3
pylint	https://www.pylint.org/	GPL-2.0	Python 2 / Python 3

Overview Page: https://wiki.dlr.de/display/SoftwareEngineering/Code+Review

Questions?

Image: © 2020 Marlene Brüggemann



Towards Automated, Provenance-driven Security Audit for git-based Repositories—Applied to Germany's Corona-Warn-App

Vision Paper

Tim Sonnekalb
Thomas S. Heinze
German Aerospace Center (DLR),
Institute of Data Science
Jena, Germany
tim.sonnekalb@dlr.de
thomas.heinze@dlr.de

Lynn von Kurnatowski
German Aerospace Center (DLR),
Institute for Software Technology
Weßling, Germany
lynn.kurnatowski@dlr.de

Andreas Schreiber
German Aerospace Center (DLR),
Institute for Software Technology
Köln, Germany
andreas.schreiber@dlr.de

Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos
Fuenlabrada, Spain
jgb@gsyc.urjc.es

Heather Packer
University of Southampton
Southampton, United Kingdom
hp3@ecs.soton.ac.uk

ABSTRACT

Software repositories contain information about source code, software development processes, and team interactions. We combine provenance of the development process with code security analysis to automatically discover insights. This provides fast feedback on the software's design and security issues, which we evaluate on projects that are developed under time pressure, such as Germany's COVID-19 contact tracing app 'Corona-Warn-App'.

KEYWORDS

program analysis, provenance, software security, repository mining, open source software, covid-19

ACM Reference Format:
Tim Sonnekalb, Thomas S. Heinze, Lynn von Kurnatowski, Andreas Schreiber, Jesus M. Gonzalez-Barahona, and Heather Packer. 2020. Towards Automated, Provenance-driven Security Audit for git-based Repositories—Applied to Germany's Corona-Warn-App: Vision Paper. In *SEAD 2020: 3rd International Workshop on Software Security from Design to Deployment, November 9, 2020, Sacramento, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Software repositories contain much information besides the source code itself. Especially for Open Source projects, the team composition and development process is transparent and traceable and can be evaluated at any point of time by, for example, continuous evaluation with regards to security by automated analysis [9].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SEAD 2020, November 9, 2020, Sacramento, CA, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/19/06…$15.00
https://doi.org/10.1145/1122445.1122456

The COVID-19 pandemic [3] raises challenges for scientists of many disciplines. Computer scientists and software developers help to fight the pandemic with software systems, which must be developed under time pressure [2], with high quality, and with accepted concepts for data security and privacy.

For example, apps for mobile devices that support *contact tracing* of infected persons are useful to identify local COVID-19 hot-spots and find other persons, who are potentially infected, too. For contact tracing, several architectures are possible and have been discussed—sometimes very controversial—in many countries. Two favoured approaches are centralized and decentralized architectures; both using Bluetooth Low Energy for contact identification. Apple and Google developed an *Exposure Notification API*[1] as extension of their operating systems iOS and Android, which developers of exposure notification apps can use for privacy-preserving contact tracing. We focus on the German decentralized exposure notification app *Corona-Warn-App* (CWA; see Section 2).

Our main contributions towards our vision of an *automated, provenance-driven security audit infrastructure for Open Source software* are:

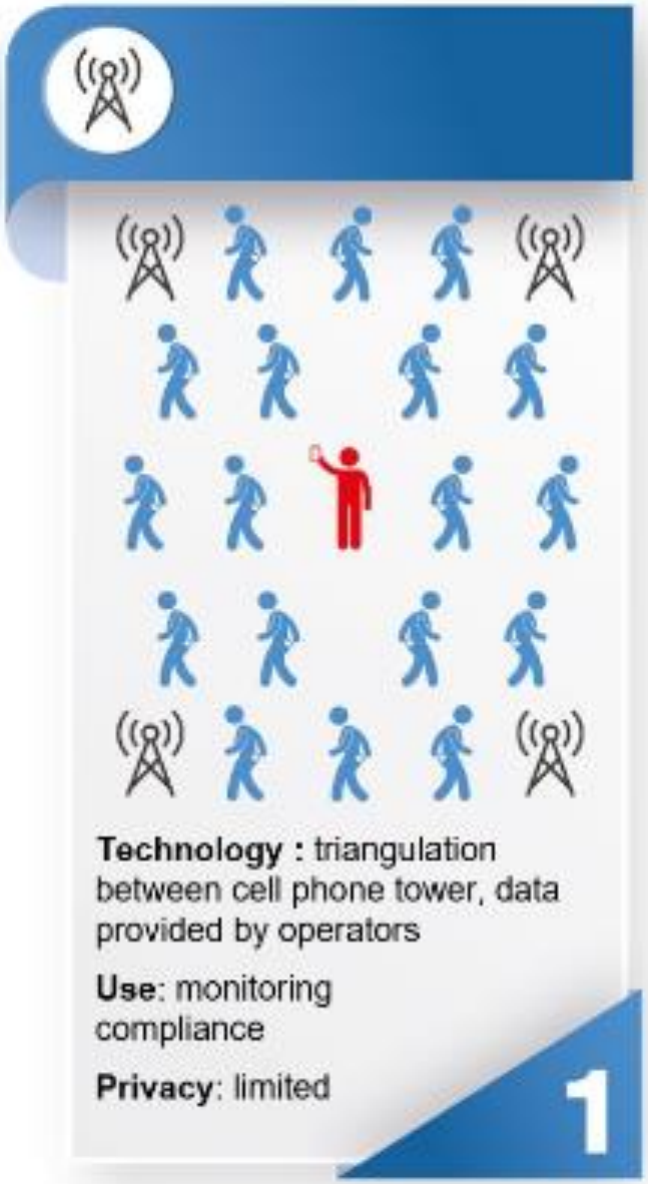
- We give an overview of static code analysis, which we use for our purpose (Section 3).
- We describe our method for querying the development process by using provenance (Section 4).
- We outline our ongoing efforts on combining information from process provenance with static code analysis for some specific revisions of the source code (Section 5).

2 DEVELOPMENT OF THE "CORONA-WARN-APP"

The development of the Corona-Warn-App gets special attention during the COVID-19 pandemic; the development had to be done in a short time frame: development started in April 2020 and the app was released on 16th June, 2020 for Android and iOS. CWA is developed by SAP and Telekom using a transparent and open

[1] https://www.apple.com/covid19/contacttracing/

BACKUP

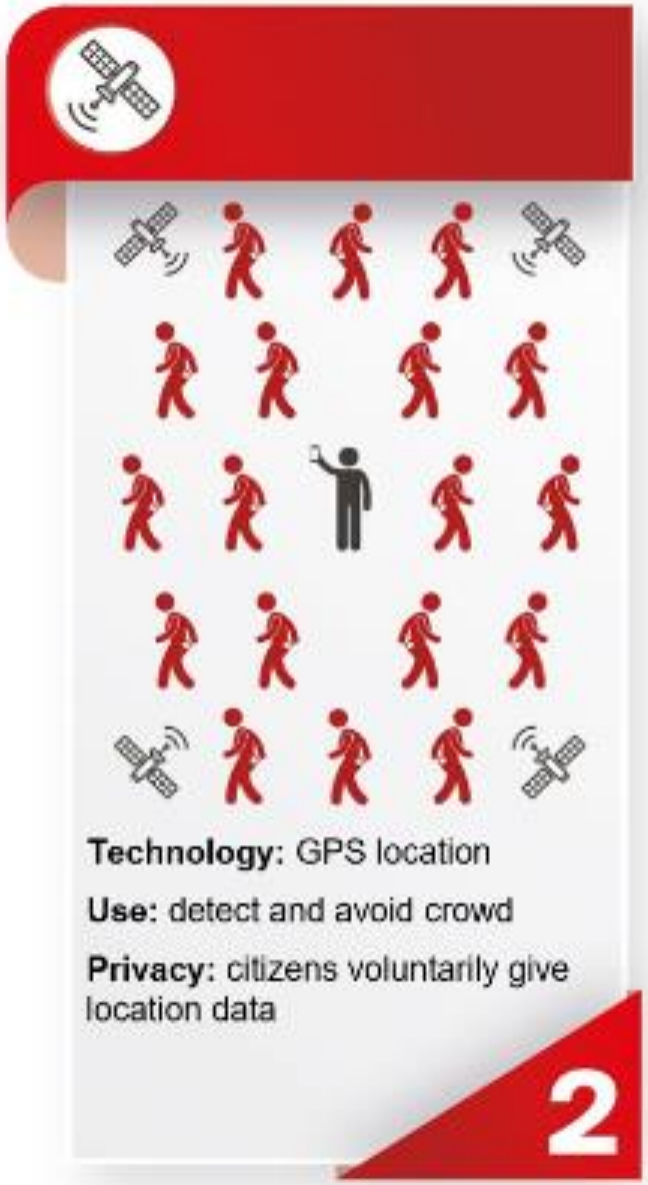


Technology : triangulation between cell phone tower, data provided by operators

Use: monitoring compliance

Privacy: limited

1

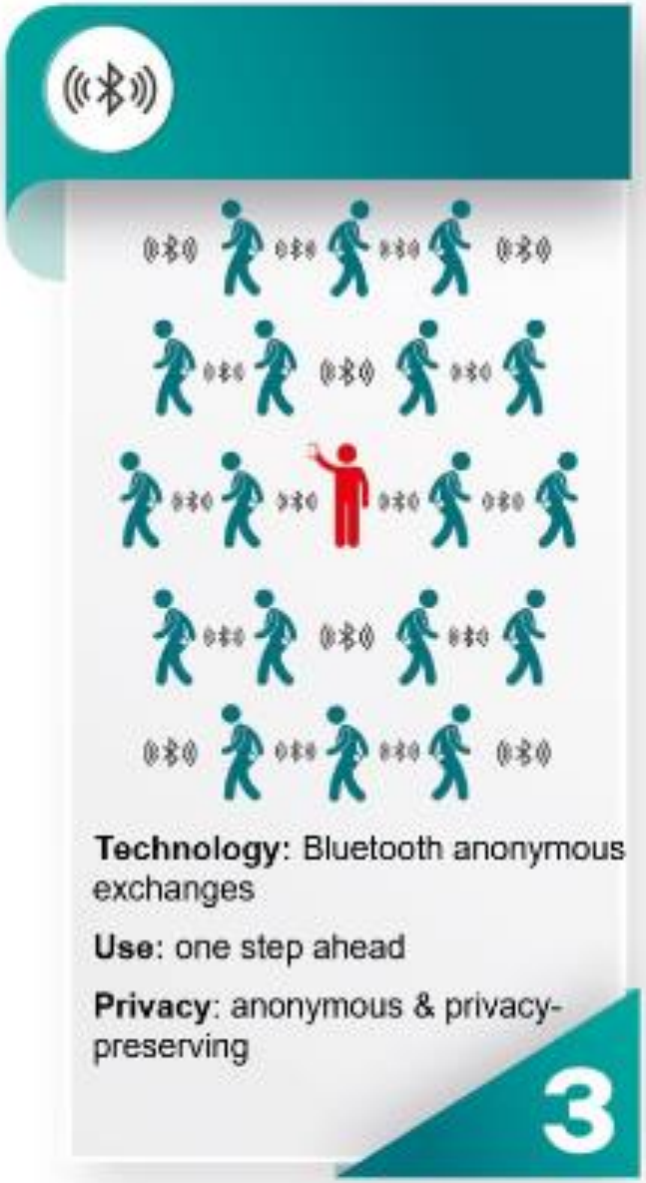


Technology: GPS location

Use: detect and avoid crowd

Privacy: citizens voluntarily give location data

2



Technology: Bluetooth anonymous exchanges

Use: one step ahead

Privacy: anonymous & privacy-preserving

3