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

WAW SE, 10.09.2020
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](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?
HOW PRIVACY-FIRST CONTACT TRACING WORKS

Alice's phone broadcasts a random message every few minutes. Alice sits next to Bob. Their phones exchange messages.

Both phones remember what they said & heard in the past 14 days. If Alice gets Covid-19, she sends her messages to a hospital.

WHAT I SAID
- 51Pomk
- 8jUI4
- 11wda6

WHAT I HEARD
- 51Pomk
- 8jUI4
- 11wda6

WHAT COVID-19 CASES SAID
Because the messages are random, no info's revealed to the hospital...

...but Bob's phone can find out if it "heard" any messages from Covid-19 cases!

If it "heard" enough messages, meaning Bob was exposed for a long enough time, he'll be alerted.

And that's how contact tracing can protect our health and privacy!

by Nicky Case (necast.me). CC0/public domain, feel free to re-post anywhere!

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

- 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

Formula used to calculate the Exposure Risk Value - totalRiskScore

\[
\text{Transmit Risk Value} \times \text{Duration Risk Value} \times \text{Days Risk Value} \times \text{Attenuation Risk Value} = \text{Exposure Risk Value}
\]
Corona-Warn-App

- Germany’s official *Exposure Notification App* ([www.coronawarn.app](http://www.coronawarn.app))
- Developed by SAP and Deutsche Telekom

- Open source software on Github ([https://github.com/corona-warn-app](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
„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

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)-[:CONTRIBUTES_TO {role: 'team'}]->(f:Entity)<-[:CONTRIBUTES_TO {role: 'contributor'}]->(external_contributor:Agent)

RETURNS
  team_member, f, external_contributor

„Are changes from external contributors possibly more insecure?“
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

Overview Page: https://wiki.dlr.de/display/SoftwareEngineering/Code+Review
Towards Automated, Provenance-driven Security Audit for git-based Repositories—Applied to Germany’s Corona-Warn-App

Vision Paper

Tim Sonnekohl
Thomas S. Herrme
German Aerospace Centre (DLR),
Institute of Data Science,
Jena, Germany
tim.sonnekohl@dlr.de
thomas.herrme@dlr.de

Lynn von Kamatskow
German Aerospace Centre (DLR),
Institute for Software Technology,
Weßling, Germany
lynn.kamatskow@dlr.de

Jesús M. González-Romobona
University of Seville
Seville, Spain
jg@jgpyc.us.es

Andreas Schürcher
German Aerospace Centre (DLR),
Institute for Software Technology,
Köln, Germany
andreas.schuercher@dlr.de

Heather Padavan
University of Southampton,
Southampton, United Kingdom
h.padavan@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 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 tight pressure, such as Germany’s COVID-19 contact tracing app “Corona-Warn-App.”

KEYWORDS
software security, source code analysis, security audit, repository analysis, open source software, covid-19

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 regard to security by automatic analysis [9].

Provenance can be defined as a snapshot of the state of a project at a specific point in time. Information about how the state of the system was reached is called provenance. In the context of software systems, provenance is defined as a complex system of dependencies between the project’s individual files. When a project is developed, each commit contains a context of the project state before and after the commit. This context is described by the provenance of the project. Provenance can be used to automatically derive information about the development process and can be used for various purposes, such as detecting potential security vulnerabilities.

2 DEVELOPMENT OF THE "CORONA-WARN-APP"
The development of the Corona-Warn-App gets special attention during the COVID-19 pandemic; the development has to be done in a short time frame. Development started in April 2020 and the app was released on 10th June 2020 for Android and iOS. COVID-19 is developed by SAP and Tabeos using a transparent and open-source approach.

Questions?
BACKUP
1. **Technology**: triangulation between cell phone tower, data provided by operators
   **Use**: monitoring compliance
   **Privacy**: limited

2. **Technology**: GPS location
   **Use**: detect and avoid crowd
   **Privacy**: citizens voluntarily give location data

3. **Technology**: Bluetooth anonymous exchanges
   **Use**: one step ahead
   **Privacy**: anonymous & privacy-preserving