Risk Estimation and Recommendations for Swiss Proximity Tracing App

Author: csirt@bit.admin.ch & outreach@govcert.ch
Topic: Risk Estimation and Recommendations for Proximity Tracing App
Date: 25th of May 2020
Classification: TLP WHITE

Introduction

CSIRT FOITT and GovCERT.ch have tested all components of the Swiss Proximity Tracing System during several weeks. We have accompanied the involved development teams during this process while continuously reporting bugs and suggesting risk mitigating measures. We believe that the application as a whole has reached a good state in terms of security and privacy. The whole proximity tracing system consists of the following components:

COMPONENT	DESCRIPTION
RED FRONTEND	The RED frontend is basically the app that is interacting with the user and the Google/Apple API in order to do the proximity tracing.
RED BACKEND	The RED backend is the server-side component that is used to receive information and distribute information about an infection person by distributing their seeds.
BLACK FRONTEND	The BLACK frontend is interfacing with medical personnel that confirms an infection and issues an authentication code that needs to be sent alongside the seeds and by doing so confirming an actual infection message.
BLACK BACKEND	The BLACK backend is the server-side component that is used to issue authentication codes.
SUPPORTING SYSTEMS	The most important supporting systems are the authentication components for the BLACK system. There are two, one operated by HIN (Health Information Network) and eIAM operated by FOITT. Other supporting systems are website but also underlying components such as operating systems, database server, monitoring components, etc..

Protocol

We consider the underlying protocol designed by EPFL to be very robust and well thought-out and believe that this is the right approach for proximity tracing, especially the chosen *decentralized approach*. We have identified a potential demasking of users; but with the addition of *fake POST requests*, this risk has been reduced to an acceptable level. By using fake POST requests, a concern about the anonymity of a user uploading an infection message has been eliminated. There remains a residual risk in case a user mistypes the authentication code while his network traffic is being monitored. The approach proposed by EPFL by adding a 13th *check digit* to the authorization code computed using Luhn algorithm would reduce the risk for many cases while – as outlined by EPFL – cannot cover all cases when users are making typing errors. We believe that it would be a fair tradeoff between ease of implementation and security/privacy. The project however decided against the implementation of check digits.

Architecture

When it comes to the technical architecture, we believe the current usage pattern of a CDN (*Content Delivery Network*, in our case it is Amazon) shows a reasonable balance between privacy and resilience. The potentially more delicate POST requests are sent directly to the FOITT backends, whereas the less sensitive GET requests are passing the CDN. By doing a *pinning of the certificate,* a near end-to-end encryption is reached and should prevent all attempts of man-in-the-middle attacks, either by companies' proxy servers doing TLS inspection, or by a malicious actor sitting between the user and the end points. We have suggested to *additionally encrypt the more delicate POST parameters* (seed of the infected device, resp. daily keys) *asymmetrically*, which is not realized at the moment. As a mitigating measure, however, pinning is done on the certificate level, i.e. the traffic in transit cannot be broken, as the app would notice this. One has to be aware that a certain flexibility in the network topology is given up by this approach.
On the side of the backends, a web application firewall is in place mitigating risks further. In case of a DDoS attack forcing the usage of the CDN for POST requests as well, a TLS interception on the CDN level should not be made as it would risk the exposure of infected users to the CDN operators.

Code

The code analyzed is well written and the architecture was developed with a security point of view (choice of communication channels, use of WAF). Most vulnerabilities were found in supporting systems and not in the core systems. We reported these vulnerabilities and besides very few, most are already fixed.

The source code of the iOS app and Android app was partially reviewed and looks well-structured and programmed. We did not find any big issues and bug reports have been dealt with quickly. As currently there is a switch between the previous version that interacted directly with the Bluetooth stack and the APIs provided by Google and Apple, we tested both approaches, but the first one more thoroughly. The testing of the backend systems did not reveal any critical vulnerability from within the code, but mostly parametrization issues. These were resolved quickly.

Apart from the app, the most complex part lies within the BLACK system, which enables authentication via eIAM and via HIN. This is where the medical staff create the authentication codes. This system is much more complex than the RED backend and has more exposed interfaces. However, the main risk here is not so much about leaking personal data, but rather that someone can obtain false Auth Codes and thus smuggle false reports of infections into the system.

There are a few security concerns that lie outside the scope of this document but should be mentioned, nevertheless. One is the overall security of the smartphone, such as revealing the identity by the name of the device ("Max Muster's Iphone") or by outdated OS versions with known vulnerabilities, especially in the Bluetooth stack. Another noteworthy risk are the devices of medical staff. If such a device gets infected, an attacker might generate authentication codes and could potentially flood the system with wrong infection data.

There are still a few potential and purely functional issues of the code or underlying API that we noticed and forwarded, but these are not security related.

Risk Estimation and Recommendation

The following table should give a short overview of our perception of risks:

Item	Residual risk	Remarks
Protocol	Low	Most possible attacks are avoided or mitigated. We would have wished having a "real" end-to-end encryption, but certificate pinning is in place and reduces the attack surface. Another point that remains is the possibility of detecting a user that enters the Auth Code wrong and then corrects it.
Architecture	Low	The architecture does not have any critical issues. The usage of CDN makes sense. If – due to a DDoS – CDN is also needed for POST requests, no TLS interception must be done on CDN Level.
Backends	Low	There are no remaining, critical vulnerabilities left open. Due to the complexity of the black backend, we see a larger risk exposure there, but currently have not seen any vulnerability left open on the core system/application.
Apps	Low	The apps are well developed and behave as expected, cryptography, communication and error handling are done correctly.
Supporting systems	Medium	Most vulnerabilities have been found in supporting systems. As some of these are either rather aged and/or highly complex, we believe that there is a certain likelihood that more vulnerabilities could be discovered there.

We recommend to closely monitor all relevant logfiles during the public security test in order to detect any intrusion by a malicious actor and to completely reset the system after the public security test. It is likely that researchers are going to find additional vulnerabilities that need to be addressed and patched. It is important to understand that this is the goal of such a test and does not mean a failure of the system and application as long as no unfixable issues are found, or the number of the vulnerabilities reported is so large that doubts arise about the underlying quality.

We also recommend implementing automated security tests that ensure that changes at the code base are handled carefully and that vulnerabilities are not introduced at a later point. As important as the development is the operational phase, which requires a good patch management, not only for the underlying operating system, but also for all used components, especially for libraries and frameworks. The logfiles of all systems need to be analyzed by CSIRT on a regular base. We also recommend talking with the Identity Provider HIN about their strategy of logfile monitoring as the authentication of medical persons is an important part of the whole security architecture.

At the point of writing this report, there are a few minor security issues still open, but nothing that would prevent a public security test. We believe that there is a good chance that researchers are going to make findings, but we are confident that none of these would put either the backend infrastructure or a user's privacy in danger.

incidents@
govcert.ch

Digital unterschrieben
von
incidents@govcert.ch
Datum: 2020.05.27
11:58:05 +02'00'