



REVIEW FACILITY.EU

Name: NGI Emergency Tech Review Facility High-level assessment BlueTrace

Contract nr: LC-04199045

Date of signature: 15-01-2021

Name of the deliverable: Deliverable 3.1 – Quick security evaluation BlueTrace

Authors: Abhinav Mishra, Stefan Marsiske (Radically Open Security)

Level of distribution: Public

Confidential: No



Quick Security Evaluation

Bluetrace

Emergency Tech Review Facility

V 1.0

Diemen, June 5th, 2020

1 Introduction

As part of the [Emergency Tech Review Facility](#) your project **Bluetrace** has received a basic quick security evaluation from Radically Open Security. The goal of the review is to provide advice and input to consider in the further development of your project.

The selected project gets 1.5 persondays from ROS for a quick security evaluation. Note that - particularly for flagging implementation issues - this is too short a period to perform a thorough check of the application. We encourage that a more comprehensively scoped in-depth test of the app be performed after reading the concerns noted in this short report.

1.1 Emergency Tech Review Facility

The Emergency Tech Review Facility is [an initiative of the European Commission](#) to independently review and assess technologies developed to fight COVID-19. This review facility is intended as a collaborative, community-focused effort to quickly and transparently analyse solutions brought forward for their applicability, security and privacy characteristics, and to subsequently fast-track the maturity of the most promising technologies.

1.2 Bluetrace

Bluetrace claims to be a decentralized/centralized contact tracing protocol: the collection of contacts, which is based on BLE (Bluetooth Low Energy), is the decentralized part (in contrast to centralised collection based on e.g. telco tracking). The contact tracing is the centralized part.

Healthcare authorities (HA) map phone numbers to random user IDs, then hand out temporary IDs every 15 minutes (in batches and in advance, in case someone goes offline). A temporary ID looks as follows:

```
AES256GCM(K_HA, IV (random), "userid (21B) | validfrom (4B) | validtill (4B)") | IV (16B) | Auth Tag (16B)
```

The HA secret key looks like a singleton, meaning that anyone who gets hold of it can decode all temporary IDs. This, in combination with the random user ID/phone number mapping, makes for a juicy target. Especially if a list of infected temporary IDs of physical contacts also exists.

Randomized BLE MAC addresses are only listed as a nuisance, which causes problems, as identical beacons are collected multiple times from the same device (which looks like a different one due to the random MAC address). Random MAC addresses are a privacy tool, not a nuisance.

The encountered messages are in UTF-8 JSON format and contain the temporary ID, the phone model, the relevant HA ID and version, and (in responses) also an RSS value. All in all this amounts to more than 128 bytes being sent in one direction. It also means that Bluetrace needs to establish a connection between the two devices and exchange more

than one packet, which increases the battery power cost of the protocol and thus is less efficient than protocols which only need to broadcast and scan for beacons.

Upon infection, victims are asked to upload their logs of encounter messages to the HA, which decrypts them and calls people on their phone to verify and validate the severity of the contact.

2 Bluetrace Security Evaluation

2.1 Tasks Performed

- Scanned the repository at <https://github.com/opentrace-community/opentrace-android> and <https://github.com/opentrace-community/opentrace-ios>
- Analysed the mobile application's (iOS and Android) code for security issues
- Analysed the mobile applications for missing best security practices
- Analysed the protocol whitepaper and existing literature on centralized contact tracing systems

2.2 Security Considerations

2.2.1 Architectural

General

The DP3T project has written a generic document which has specific sections on centralized systems that also apply to Bluetrace: [https://github.com/DP-3T/documents/raw/master/Security analysis/Privacy and Security Attacks on Digital Proximity Tracing Systems.pdf](https://github.com/DP-3T/documents/raw/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf)

Susceptibility to location confirmation attacks

If a totalitarian state intervenes with the Healthcare authority (HA) and provides the centrally allocated identifiers to LE, or creates specific, easy to distinguish identifiers for persons of interest (PoI), it's would be only a matter of placing some cheap bluetooth-enabled devices at locations of interest (LoI) to have a very inexpensive and accurate method of verifying that some PoI is present at an LoI. With decentralized systems, such as DP3T or the Google/Apple protocol, this is not possible. In failed authoritarian states, which we have in the EU as well, this can be a serious threat – think e.g. of opposition leaders, and of police riot extraction teams targeting these people specifically at demonstrations.

Key management issues

The HA maintains a database that links phone numbers (to contact people) to unique identifiers; these identifiers are encrypted with one key, which is also held by the HA. No health authority in the world has the competences or the security stance to properly manage keys like these or to protect a database of this kind to a reassuring degree. Outsourcing the task to a third party would give power over the key and the database to the third party. This is a serious public trust issue which cannot be reconciled.

When someone is tested positive, they are encouraged to upload all captured Bluetrace beacons (which are basically encrypted IDs linked to phone numbers) to the HA, whose epidemiologist contact tracers can then call all people the this person encountered. This means:

- That the HA must be able to decrypt the beacons (so it needs access to the key mentioned in the previous point) and that the HA is in possession of all tracked social encounters made by the infected person. In the case of failed totalitarian states - which we have in the EU too - this means that it might be possible to connect a whistleblower contact to a journalist, or to track opposition groups. This is cheap, can happen without a warrant (which *is* needed for cellphone tracking) and can happen after the fact (but within 14 days, or whatever the retention period of Bluetrace beacons is).
- If for lack of trust in the data protection competences of the health authority the key is outsourced to a 3rd party, then the attack surface is extended to that 3rd party too, and everything noted in the previous sub-point would still apply.

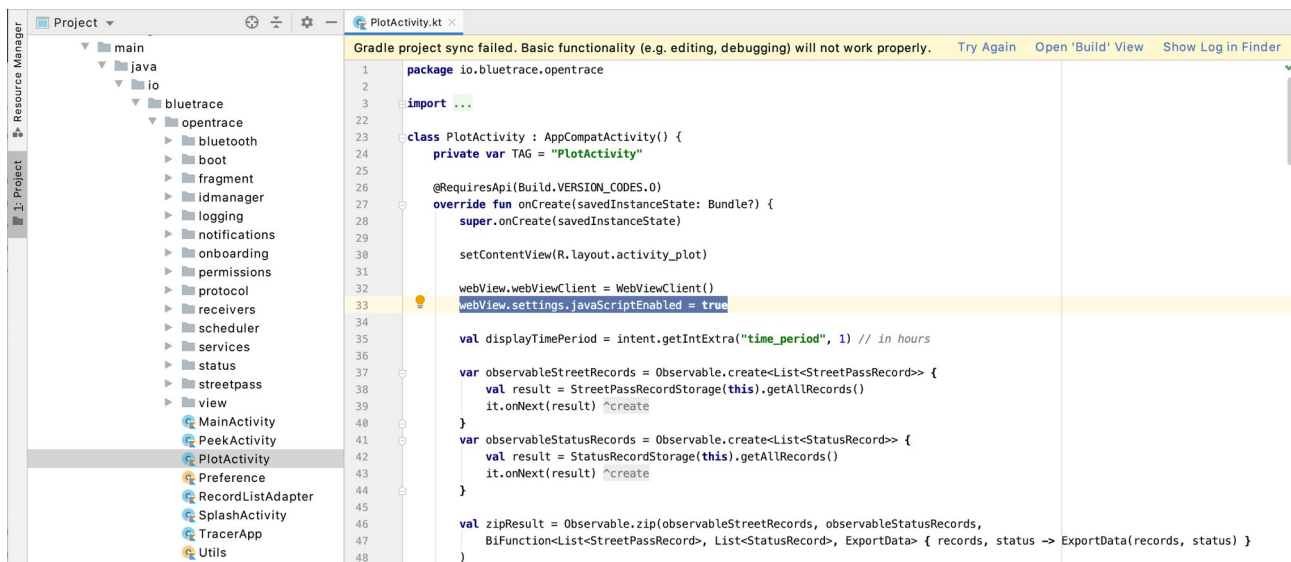
Societal impact

Anyone in opposition to the state in any way is discouraged from using this app, as it can be used against them. This creates a two-class society: those who are in-line with the state are protected by this app, and those who are opposed to it are more exposed due to reduced protection. Bluetrace can be turned a tool of oppression.

2.2.2 Implementation

Misconfiguration

The configuration for `webView.settings.javaScriptEnabled` is set to `true` in `PlotActivity`. Unless javascript is required needed to run for the app to work, it should be set to `false`.

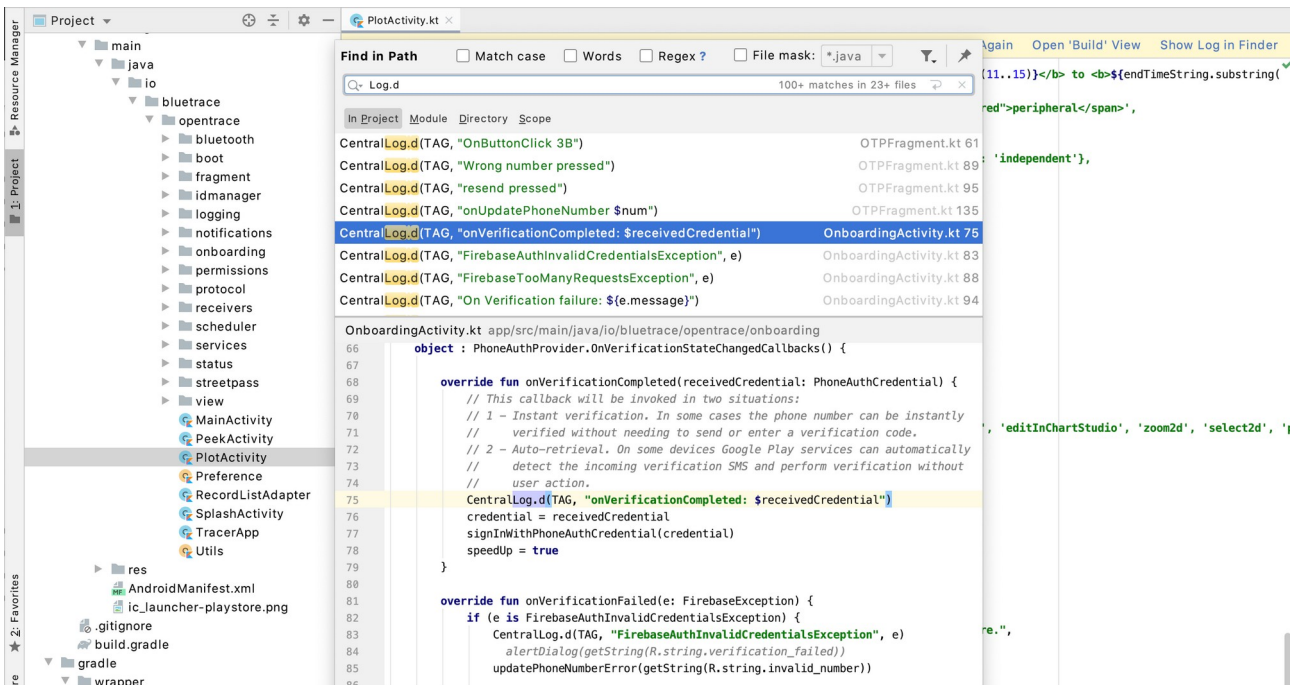
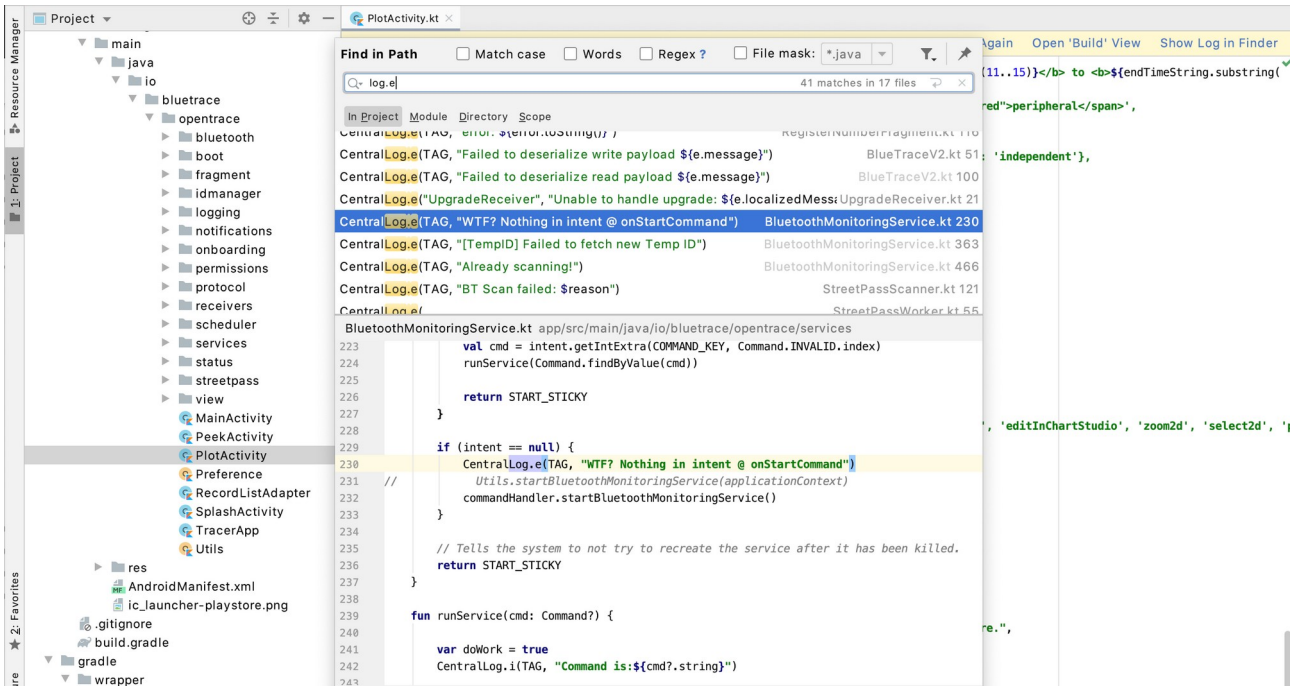


```
1 package io.bluetrace.opentrace
2
3 import ...
4
5 class PlotActivity : AppCompatActivity() {
6     private var TAG = "PlotActivity"
7
8     @RequiresApi(Build.VERSION_CODES.O)
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_plot)
12
13         webView.webViewClient = WebViewClient()
14         webView.settings.javaScriptEnabled = true
15
16         val displayTimePeriod = intent.getIntExtra("time_period", 1) // in hours
17
18         var observableStreetRecords = Observable.create<List<StreetPassRecord>> {
19             val result = StreetPassRecordStorage(this).getAllRecords()
20             it.onNext(result) ^create
21         }
22
23         var observableStatusRecords = Observable.create<List<StatusRecord>> {
24             val result = StatusRecordStorage(this).getAllRecords()
25             it.onNext(result) ^create
26         }
27
28         val zipResult = Observable.zip(observableStreetRecords, observableStatusRecords,
29             BiFunction<List<StreetPassRecord>, List<StatusRecord>, ExportData> { records, status -> ExportData(records, status) }
30     )
31 }
```

webView.settings.javaScriptEnabled set to true

Sensitive information disclosure

Multiple entries for logcat were discovered in the app's code, Log.d, Log.e etc. Disclosing sensitive information in Android's logcat is not recommended.



Data logging in external storage

The application logs data in ExternalStorageDirectory. Files in external storage might be accessible to other apps installed on the same device.

```

private fun createFileWriter(dateStamp: String): BufferedWriter {
    val dir = File(Environment.getExternalStorageDirectory().absolutePath + "/" + FOLDER)
    dir.mkdirs()
    val file = File(dir, APP_NAME + "_" + dateStamp + ".txt")
    val fw = FileWriter(file, true)
    return fw.buffered()
}

private fun getFileWriter(): BufferedWriter {
    //date stamp for filename
    val dateStamp = dateFormat.format(Date())

    return if (dateStamp == cachedDateStamp) {
        cachedFileWriter
    } else {
        //make sure all the logs from previous day is written to the previous file
        if (::cachedFileWriter.isInitialized) {
            cachedFileWriter.flush()
            cachedFileWriter.close()
        }

        //create a new fileWriter for the day
        cachedFileWriter = createFileWriter(dateStamp)
        cachedDateStamp = dateStamp
        cachedFileWriter
    }
}
}

```

Logging to ExternalStorageDirectory

2.3 Recommendations

While this evaluation was done in a mere 1.5 days and is in no way to be considered a full audit or pentest, based on our short quickscan of the Bluetrace app and protocol we can already recommend the following:

Architectural

- Discourage the deployment of Bluetrace or similar centralized protocols, as they can be abused by for instance totalitarian states.

Implementation

- `webView.settings.javaScriptEnabled` should be set to false.
- Ensure that the application does not send any sensitive user/application data in logs. All the logcat entries should be analysed and only allowed if they are needed for the application to work.

- Storing user-sensitive data in external storage is not recommended; only use the internal application's storage to store user data.

3 Contact details

Radically Open Security is the world's first not-for-profit computer security consultancy. We operate under an innovative new business model whereby we use a Dutch fiscal entity, called a "Fiscaal Fondswervende Instelling" (Fiscal Fund raising Institution), as a commercial front-end to send 90% of our profits, tax-free, to a not-for-profit foundation, Stichting NLnet. The NLnet Foundation has supported open-source, digital rights, and Internet research for almost 20 years.

In contrast to other organizations, our profits do not benefit shareholders, investors, or founders. Our profits benefit society. As an organization without a profit-motive, we recruit top-name, ethical security experts and find like-minded customers that want to use their IT security budget as a "vote" to support socially responsible entrepreneurship. The rapid pace of our current growth reflects the positive response the market has to our idealistic philosophy and innovative business model.

If you have any questions about the advice in this report, please contact us at info@radicallyopensecurity.com

For more information about Radically Open Security and its services please visit our website:

www.radicallyopensecurity.com.

The pentesters for this project were:

- Abhinav Mishra
Abhinav has 9+ years of experience in penetration testing of web, mobile, infrastructure, APIs etc. He has received numerous accolades from multiple organizations for responsible disclosure of vulnerabilities. He is also known for providing trainings on web, mobile and infrastructure security.
- Stefan Marsiske
Stefan runs workshops on radare2, embedded hardware, lock-picking, soldering, gnuradio/SDR, reverse-engineering, and crypto topics. In 2015 he scored in the top 10 of the Conference on Cryptographic Hardware and Embedded Systems Challenge. He has run training courses on OPSEC for journalists and NGOs, he played a lot with gnupg and is the maintainer of pysodium.

4 Disclaimer

It is important to understand the limits of ROS' services. ROS does not (and cannot) give guarantees that something is secure. Additionally, the above advice is obviously incomparable to a full-blown security audit as performed by ROS or any other professional security company, which takes orders of magnitude more time and effort. (Such an audit may be necessary in the future still, and we would be happy to work with you on that).

We recommend for now you treat our feedback as a discreet sanity check by knowledgeable friends; it is not the intention to publicly claim that Radically Open Security audited your code and found it to be safe or 'found no problems'. Rather, the intention is the reverse: we aim to help you capture obvious flaws within the very limited terms of this deal, and to protect the general audience from irresponsible projects putting them at risk.

There is no shame in clearly messaging to users that your project is still in an early stage, that they use it at their own risk and that there are no guarantees - being honest with your users can only ever be a good thing.

Finally, we want to emphasize that security is a process – this security evaluation is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your information security. We hope that this report will contribute meaningfully towards that end.