TECHNICAL AND

SECURITY SPECIFICATIONS

Tableau Extensions

Document Information

Date:	12 March 2020
Version:	1.0
Version Release Date:	12 March 2020

Table of Contents

1 Technical Specifications

1.1 Overview

This document provides general high-level information about architecture and technical aspects of how Tableau extensions work and handle data.

1.2 Requirements

Tableau extensions are supported on Tableau Desktop, Tableau Server, and Tableau Online. Minimum Tableau version requirements for using extensions:

- Tableau Desktop 2018.2 and later
- Tableau Server 2018.2 and later
- Tableau Online

1.3 What are Tableau extensions?

Tableau extensions are essentially web applications that run inside Tableau dashboard(s) and allow for two-way communication between extension, data imported in Tableau and Tableau objects (filters, parameters, settings…).

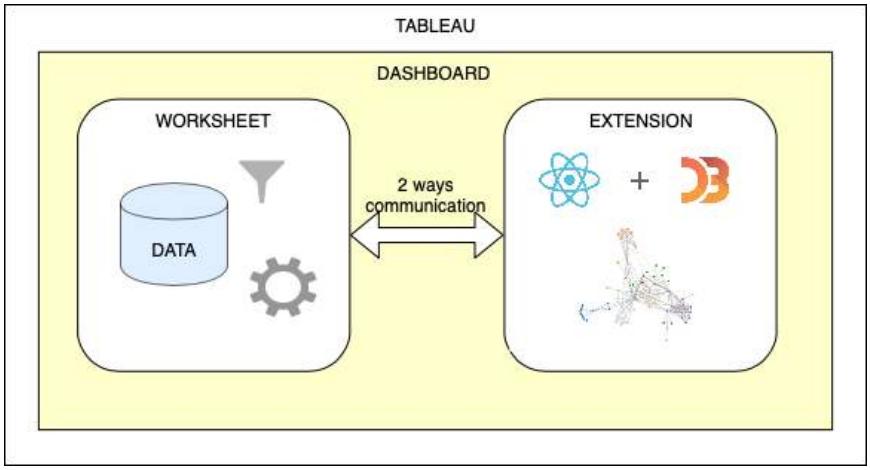
By default, or unless stated otherwise, each extension provides 14 days free trial of its functionality. After extension expires, user needs to obtain a license from Billigence to continue using it.

1.4 Technologies

Different technologies were used to build Tableau extensions provided by Billigence:

- **Tableau extensions API**: open source library from Tableau for extensions development
  https://tableau.github.io/extensions-api
- **ReactJS**: a declarative, efficient and flexible JavaScript library to build user interfaces
  https://reactjs.org
- **D3.js**: a JavaScript library for producing dynamic, interactive data visualizations
  https://d3js.org
- **Bootstrap**: a front-end framework to develop user interfaces
  https://getbootstrap.com
- **AdonisJs:** stable, simple to use Node.js framework
  https://adonisjs.com
- **Microsoft Azure:** Extensions run in the cloud. Microsoft Azure offers first class services to build secure and scalable web applications
  http://azure.microsoft.com

Coding extensions using React and D3.js allows for great flexibility in building complex data visualizations, as well as create first class user experience. Utilising Tableau extensions API then makes it possible to work with data in Tableau and have the extension react to changes and / or updates in Tableau dashboard(s) due to e.g. change of filters etc.



1.5 What is .trex file?

Each Tableau extension consists of a manifest file (.trex). These manifest files contain metadata instructions on how to connect to respective extension in the cloud, there's no code or logic written inside of them. They're also the only thing needed to start using extensions within Tableau environment. By placing extension into dashboard(s) as object, a user initiates the follow-up authentication phase, which is critical in order to start using the extension.

Individual .trex files for respective extensions can be downloaded via Billigence website. For more information on how to add Tableau extensions into dashboard(s), please visit the following link: *https://tableau.github.io/extensions-api/docs/trex_getstarted.html*

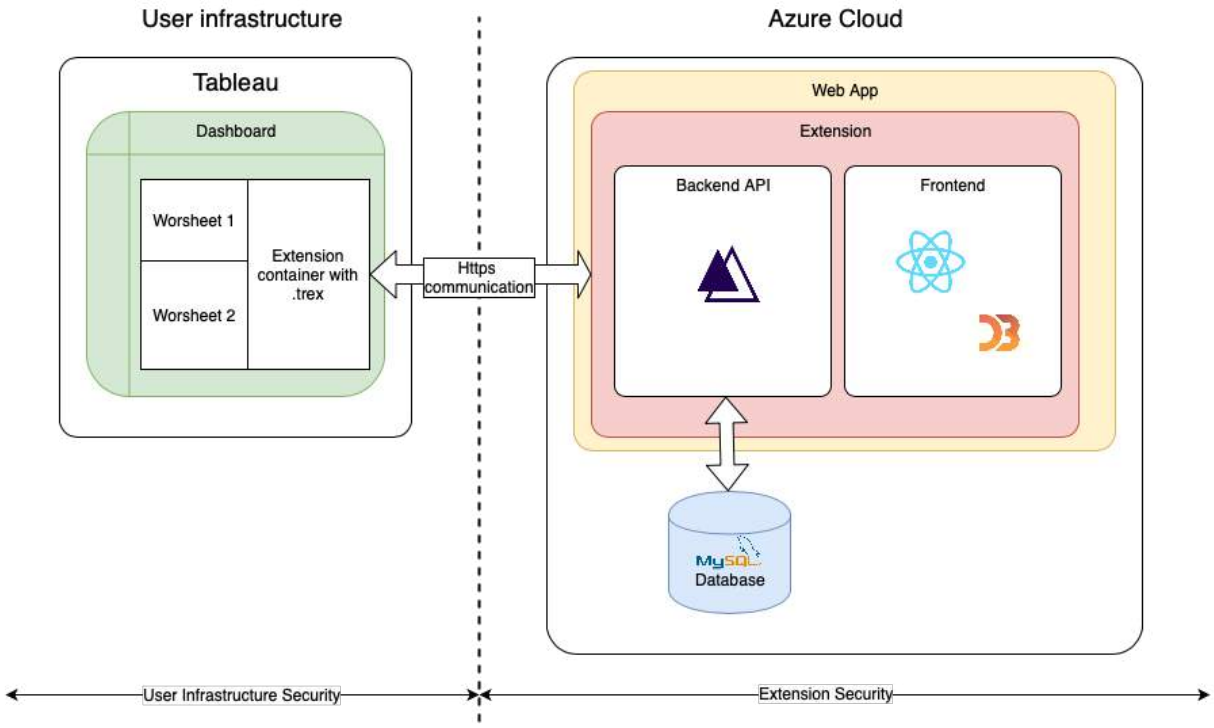
1.6 Security measures

Tableau extension design consists of three layers:

- Frontend
- Backend API
- Database

All three parts reside in the cloud and it is the .trex manifest file that allows user to connect to the above extension ecosystem from Tableau.

The below scheme and information details all measures taken to ensure maximum data security across all endpoints within the environment.



By placing the .trex manifest file as object into Tableau dashboard(s), user allows it to point to a remote domain of respective web application on Microsoft Azure. This provides access to extension's interface, which always begins with user authentication.

When using extension for the first time, the user has to register and create credentials to be used to access the extension at any later time. User data is securely stored in the database (not part of Microsoft Azure) with backend API managing communication between the database and extension's authentication interface. Successful authentication then allows user to start using the extension.

User Infrastructure Security: Billigence is in any case not responsible for security within user premises (left side of above scheme). It is sole responsibility of the user and their governance / data security team to guarantee that unknown users cannot gain access to or use their systems including Tableau and data within it.

Extension Security: is primarily based on security measures of Tableau extensions API, Microsoft Azure cloud infrastructure and third-party libraries / frameworks used to develop extensions. On top of that great effort was gave to be sure that mostly of security standards were applied on each of the components involved. These points are described on the next chapter.

Trex

- All .trex files are securely stored in Microsoft Azure cloud infrastructure. Both trial and licensed extension modes are managed via authentication backend with all relevant user information securely stored in the database.

Data handling

Because of their nature, extensions can interact with other components in the dashboard(s) and have access to underlying data in the workbook (via Tableau API).

Below measures were taken to guarantee safety of sensitive data:

- Only official Tableau API is used to gather data from Tableau.
- No user data is in any way stored outside of Tableau environment. External database is only used to store user authentication data.
- Connection between Tableau and extension happens via HTTPS domain created in Microsoft Azure to guarantee encrypted communication (that's where .trex files point).
- Extensions only use worksheet data part of dashboard(s) where the extension is placed and this data is only used for visualisation and rendering purposes; extensions don't ever have access to original underlying data in its entirety. None of above-mentioned data ever leaves its original environment.

Microsoft Azure

- Extensions are stored on Microsoft Azure infrastructure guaranteeing high security standards as well as service availability. Credentials and other user data used for authentication are encrypted.

Tableau extensions security

Tableau included several security features to guarantee security when using extensions API:

- All extensions must by default use the HTTPS (secure HTTP) protocol.
- For the extension to be run on Tableau Server, extension's URL must be added to a 'safe list'. This list is managed by Tableau Server site administrator. Please see Tableau extensions API for more information.
- Anyone willing to use any extension will be by default prompted to allow or deny it access to Tableau.

HTTPS

Using HTTPS permits that all HTTP data is encrypted prior to the transmission by the Transport Layer Security (TLS) and decrypted when it is received. HTTPS ensures a secure channel between the extension (client), running in the Tableau dashboard, and the web server that hosts the extension in Microsoft Azure infrastructure. The use of HTTPS provides a high level of privacy, authentication and integrity.

Using HTTPS assures that data is safe and that users are connecting to a trusted extension. Because the extension is using HTTPS, Tableau is also able to verify the identity of the server that hosts the extension, which prevents various malicious man-in-the-middle attacks that could occur if the extension were to use HTTP alone.

Tableau by default checks whether HTTPS standard is applied and if it is not, extension won't work.

Any third-party library used for developing the frontend part is referenced with HTTPS (even if not directly related to data).

Frontend

ReactJS is developed by Facebook community; this guarantees that the library is frequently updated to the best and latest security standards. Even when crucial part of application security resides on the backend, many security issues are in fact prevented on the frontend by ReactJS by default (e.g. XSS attacks or Script Injections).

Backend

AdonisJs provides high level security standards to build backend API. Below list details security features enabled on configuration:

- CORS (Cross-Origin Resource Sharing)

  Cross domain attacks are prevented on backend side restricting access from unknown domains. All such IPs are blocked.

- CSRF protection (Cross-Site Request Forgery (CSRF))

  CSRF attacks protection permits to prevent that an unknown user can perform actions on behalf of another user without their knowledge or permission. AdonisJs backend protects from CSFR attacks by denying unidentified requests.

  HTTP API requests have been protected to make sure that only the right people from the right place invoke these requests.

- Malware protection

  AdonisJs also automatically prevents other common malware attacks like XSS, Content-type Sniffing, Script Injection and other.

Database

All stored user passwords in the database are hashed. Decryption is done by AdonisJs and there is no option to retrieve them or perform any kind of general attack.

AdonisJs uses Bcrypt standard, which is considered one of best for password encryption.

1.7 Data retention

To avoid a scenario in which all chart visualizations and extension-related parameters are lost when Tableau workbook file is closed, the data retention feature was added. In this case, Tableau settings are used to store all parameters and variables needed, which gives the possibility to reload them once Tableau dashboard(s) with extension is / are opened again. This allows to see the same chart rendered as user was presented with before closing Tableau workbook.

Authentication tokens are also saved in Tableau to avoid users having to re-login every time.

Both above features happen on Tableau side only.