TABLEAU CONFERENCE
Welcome
Big Easy Data Security

Scalable Programmatic Solutions for Roux Level Security
SESSION REPEATS

Big Easy Data Security
Tues | 4:00 – 5:00 | MCCNO – L2 – 297

Big Easy Data Security
Wed | 10:15 – 11:15 | MCCNO – L2 – 204
Data level security with Tableau Desktop

Tues  |  12:30 – 1:30  |  MCCNO - L3 - 338

Data level security with Tableau Desktop

Wed  |  1:45 – 2:45  |  MCCNO - L2 - 211
Who Dat Gon Present To You?
The Triplets

Just three Arkansas boys coming down to Louisiana to have a little fun
We ain’t cookin’ Gumbo

“Roux Level Security” ain’t pronounced “Row”

New Orleans spelling would have been Reaux

Please don’t say “Re-Ay-You-Ucks” Level Security
Agenda

Row Level Security Overview

Standard Row Level Security in Tableau

Advanced Row Level Security Solutions
  Securely Filtering at Load Time
  XML Modification of Template Files
  “Live” Connecting to Web Service Data Sources
Row Level Security Overview
"All music is folk music. I ain’t never heard a horse sing"

Louis Armstrong
What is Row Level Security?

We hope you won’t rue the role that rows play in this session

We’re really talking about “Data Level Security”

To straighten it all out:

Roles in Tableau are for Permissions. They control what actions you can take in Tableau

Rows are things in a data table, that contain the actual data.

Row Level Security is blocking data someone shouldn’t see
Being “Entitled” (seems like everyone is this days)

Entitlement:
A single combination of Attributes that the Data will be filtered on

Ex. If you filter on Category, Sub-Category, and Region, any unique combination of those would be a single Entitlement

<table>
<thead>
<tr>
<th>Entitlement ID</th>
<th>Category</th>
<th>Sub-Category</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUR-BOO-N</td>
<td>Furniture</td>
<td>Bookshelves</td>
<td>North</td>
</tr>
<tr>
<td>FUR-BOO-S</td>
<td>Furniture</td>
<td>Bookshelves</td>
<td>South</td>
</tr>
<tr>
<td>FUR-CHA-N</td>
<td>Furniture</td>
<td>Chairs</td>
<td>North</td>
</tr>
</tbody>
</table>
Row Level Security – High Level Process Flow

- All Entitlements
- Filter On User
- User’s Entitlements
- Filter On Entitlements
- User’s Data
- All Data
If you want to make gumbo...

First you need to make a Roux
If you want to filter rows of data

You need some columns to filter on

Ex.
Category = ‘Furniture’ AND Sub-Category = ‘Bookshelves’ AND Region = ‘North’
Securing Data Filters in Tableau
In Tableau, when you drag a Field onto the Filters shelf, it creates a Filter.

Tableau Filters translate into a WHERE clause in the SQL query.

You can filter on a Calculated Field.
Data Source Filters are the Only Secure Filters

Data Source Filters cannot be cleared by End User

JS API has command that completely removes any “normal” filter
Published Data Sources Are Required

A Tableau Workbook can contain two types of Data Sources:

1. Embedded: Data source definition lives in the Workbook file
2. Published: Data Source definition has been published to a Tableau Server

Embedded Data Sources allow an end user with Web Edit or Download to edit Calculated Fields

Row Level Security requires Calculated Fields, so Embedded Data Sources are less secure. Use only if Web Edit / Download will always be disabled
Unbreakable Row Level Security Filters

1. User Calculation or Parameter in Calculated Field

2. Data Source Filter on Calculated Field

3. Published Data Source
Plain OI’ Row Level Security in Tableau
Standard Row Level Security

One Database on One Database Server

- All Entitlements View
  - WHERE Clause
  - Filter On User
    - User’s Entitlements
      - INNER JOIN
        - Filter On Entitlements
          - User’s Data
        - All Data View
Synchronizing Users into Tableau Server

To use User Calculations, you must create a User in Tableau Server

Several ways to Synchronize Users to Tableau Server
- REST API
- Active Directory
- LDAP
- tabcmd
User Calculations

Tableau Calculated Fields have access to the following calculations, which return for the logged-in user:

USERNAME()
FULLNAME()
ISMEMBEROF()

Example of Calculated Field:

[Username] = USERNAME()
When does Standard RLS work best?

All Data and Entitlements on a Single Database Server Machine

OR

Data and Entitlements as individual tables in an Extract

(In Tableau 2018.3+, Hyper Extracts can contain multiple tables)
You’re here to kick it up a notch
When Standard RLS don’t cut it

Entitlements come from some other system

Data is stored in one database/schema per customer on one or many machines

Multiple Data Warehouse servers

Data only from a Web Service with entitlements already built
Advanced RLS Solutions

Filtering Securely at Load Time
XML Modification of Template Files
“Live” Connecting to Web Service Data Sources
Should I implement an Advanced RLS Solution?

Not if you don’t have to!

If you can make your database support “Standard Row Level Security”, you should Weigh implementing any Advanced Solution against the effort of Standard RLS
Securely Filtering at Load Time
Entitlements From Application Layer

Application Layer

All Entitlements

Filter On User

User’s Entitlements

Database

All Data View

Filter On Entitlements

User’s Data

WHERE Clause

Tableau Viz URL

User’s Data

Application Layer

All Entitlements

Filter On User

User’s Entitlements

Database

All Data View

Filter On Entitlements

User’s Data
When would this be an appropriate solution?

If synchronizing entitlements from one system to the main data source would be impossibly hard integration. For example, if:

- System schemas are locked down
- Application Layer allows end users to change entitlements frequently
- Number of Entitlement Permutations too high to sync
- Entitlements are not stored in a relational model
- It is forbidden
Setting Filter Values at Report Load Time

Filters and Tableau Parameters can be set using URL Parameters or JS API.

Example of Setting Values using URL Parameters:

```
MyWorkbook/MyViz?Category=Furniture,Office Supplies&Username=bhowell
```

User can clear or reset Filters or Tableau Parameters via URL or JavaScript API.
Filters vs. Tableau Parameters

Filter: Sets the value(s) for a particular Dimension or Measure
  Creates a WHERE clause in the SQL

Tableau Parameter: Storage for a single value
  Can be used in a Calculated Field
  Calculated Fields become Dimensions that can be used as Filters

URL Parameter: Anything after the ? in a URL
  ?Something=Value
  Can set the Value of a Filter or a Tableau Parameter
Securely Filtering with Tableau Parameters

Tableau Parameter values can be set to any value or null by end user

Calculated Fields can have logic to validate and protect against "error cases"

IF LEN([Our Parameter]) == 20 THEN
    IF [Our Parameter] = [Some Field] THEN 1
    ELSE 0
END
ELSEIF ISNULL([Our Parameter]) THEN 0
ELSE 0
END
URL Parameters are an attack vector

**WARNING:** URL Parameters passed in the clear are not secure because the End User can change the values

Ex.

MyWorkbook/MySheet?username=rdugger&region=South

Remember, the end user can see the URL and try to send other variations

- Easy to see what the parameters stand for
- Easy to see the pattern of the values
- Easy to try obvious combinations
Securing the URL Parameters

Goal: Obfuscate / Encrypt both parameter names and values

In the Clear

MyWorkbook/MySheet?username=rdugger&region=South

Obfuscated/Encrypted

MyWorkbook/MySheet?p1=832h9hn9f8h234h943hr983&p2=234829482bksdfh348hr34j
Securing the Tableau Parameter Values

Parameter Values must be hashed / encrypted / encoded following the principle:

If you know both encoded and real values, you still cannot generate the encoded value for any other real value.

<table>
<thead>
<tr>
<th>Real Value</th>
<th>Encoded Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>South</td>
<td>234829482bksdfh348hr34j</td>
</tr>
<tr>
<td>North</td>
<td>???</td>
</tr>
</tbody>
</table>

Options:
- Hash Values with Shared Salt
- Encrypt Values with Shared Key
- Use non-sequential alphanumeric IDs
Calculated Field to Filter on Multiple Parameters

Parameters can only store one value at a time

To filter inclusively on multiple parameter values on the same field, use a CASE statement

```plaintext
CASE [Database Field]
WHEN [Parameter 1] THEN 1
WHEN [Parameter 2] THEN 1
WHEN [Parameter 3] THEN 1
...
ELSE 0
END
```
Best Practice for Large User Counts

Even if you have too many users to add every username into Tableau, you should at minimum create a user at the “tenant” / “customer” / “organization” level.

Best Practice for this scenario involves two levels of security filtering:
- `USERNAME()` function to filter at the “Tenant” level
- URL Parameters for additional levels of filtering
Best Practice Example

Ex. “Tenant” is a Customer with a unique ID:

IF USERNAME() == [Customer ID] THEN
    CASE [Some Field]
    WHEN [p1] THEN 1
    WHEN [p2] THEN 1
    WHEN [p3] THEN 1
    ...
    ELSE 0
    END
ELSEIF ISNULL([p1]) THEN 0
ELSE 0
END
Templates & XML Modification
<table>
<thead>
<tr>
<th></th>
<th>Single Database/Schema</th>
<th>Multiple Database/Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One Server</strong></td>
<td><strong>Single Data Warehouse</strong></td>
<td><strong>SaaS / Multi-tenancy</strong></td>
</tr>
<tr>
<td></td>
<td>Handled by Standard RLS</td>
<td>Change the Database name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Server address remains the same</td>
</tr>
<tr>
<td><strong>1 Workbook</strong></td>
<td><strong>Total</strong></td>
<td><strong>1 Workbook</strong> <em>Per Tenant</em></td>
</tr>
<tr>
<td><strong>Multiple</strong></td>
<td><strong>Multiple Data Warehouses</strong></td>
<td><strong>Multi-Server SaaS/Multi-tenancy</strong></td>
</tr>
<tr>
<td><strong>Servers</strong></td>
<td>Change the Server address</td>
<td>Change the Server address</td>
</tr>
<tr>
<td></td>
<td>DB Name / Schema remains the same</td>
<td>Change DB Name / Schema</td>
</tr>
<tr>
<td><strong>1 Workbook</strong></td>
<td><strong>per DB Server</strong></td>
<td><strong>1 Workbook</strong> <em>Per Tenant</em></td>
</tr>
</tbody>
</table>
## First Option: ETL to a Single Data Warehouse

<table>
<thead>
<tr>
<th></th>
<th>Single Database/Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One Server</strong></td>
<td><strong>Single Data Warehouse</strong></td>
</tr>
<tr>
<td></td>
<td>- Handled by Standard RLS</td>
</tr>
<tr>
<td></td>
<td><strong>1 Workbook Total</strong></td>
</tr>
<tr>
<td><strong>Multiple Servers</strong></td>
<td><strong>Multiple Data Warehouses</strong></td>
</tr>
<tr>
<td></td>
<td>- Change the Server address</td>
</tr>
<tr>
<td></td>
<td>- DB Name / Schema remains the same</td>
</tr>
<tr>
<td></td>
<td><strong>1 Workbook / DB Server</strong></td>
</tr>
</tbody>
</table>

|                  | **Multiple Database/Schema**                                                          |
| **SaaS / Multi-tenancy** | **Change the Database name** |
|                  | - Server address remains the same                                                     |
|                  | **1 Workbook per Tenant**                                                             |

|                  | **Multi-Server SaaS/Multi-tenancy**                                                  |
|                  | - Change the Server address                                                          |
|                  | - Change DB Name / Schema                                                            |
|                  | **1 Workbook per Tenant**                                                             |
XML Modification

Some attributes of Tableau Data Sources are only accessible in the XML of the Tableau files (TDS, TWB)

It is safe to modify the XML when changes affect the **Content**, but not the **Structure**

**Ex.**
Changing the Server Address, when the database/schema names and the table names are the same between servers
Structure vs. Content

What is Structure and what is Content?
Ex.
  Ron and Jon

Tableau files have XML structure
  The tag names and the attribute names in the tags are Structure
  The values are the Content

Extract files have structure too
  Column names and Data Types
Content must fit into the Structure

Different content could fit in any of these structures

Content too different to go into that type of structure
What exactly do you modify in the XML?

The **datasources** section is the same in TDS and TWB.

Changing *attribute values* in the **datasource** and **connection** tags.
# Known Working Modifications

<table>
<thead>
<tr>
<th>Database / Schema Name</th>
<th>Initial SQL statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Table Names</td>
</tr>
<tr>
<td>Connection Type</td>
<td>Table JOIN relationships</td>
</tr>
<tr>
<td>SSL Mode</td>
<td>Column Aliases</td>
</tr>
<tr>
<td>Authentication Type</td>
<td>Parameter Values</td>
</tr>
<tr>
<td>Text/Excel Filename</td>
<td>Stored Procedure Parameter Values</td>
</tr>
<tr>
<td>Custom SQL queries</td>
<td>Filter Values</td>
</tr>
</tbody>
</table>
Methods for Modifying the XML

Document API
Tableau has an official Python library called the Document API which performs XML Modification

Other Libraries
The tableau_tools Python library and the PowerTools SDK from Interworks have implemented a variety of further XML modifications beyond the Document API

Build it Yourself
Many customers also write their own XML Modification programs, depending on their development environments
Modifying XML: tableau_tools example

ds_file = 'Test Datasource.tds'
t_file = TableauFile(ds_file, logger_obj)
dses = t_file.tableau_document.datasources

for d in dses:
    for conn in d.connections:
        if conn.dbname == u'shared_resources':
            conn.dbname = u'prod_db_1'

        if conn.server == u'internal-db.our.lan'
            conn.server = u'prod-our.lan'

    t_file.save_new_file(u'Temp Datasource')
Changing Server, Database and Table Names
Changing Server, Database and Table Names
Web Services for Live Data
MADE OF BRASS
HIT WITH A STICK
NOT A "HORN"
MADE OF BRASS
Web Services / REST API “Live” Connection

Many organizations have highly optimized Web Services / REST APIs to access data

The requirements for connecting to these Web Services typically are:

- Communication over HTTPS
- Receiving responses in JSON/XML
- Passing parameters and/or authentication tokens, most often through POST method, for each user or report that is loaded

Web Services responses often aren’t tabular
- Need to be mapped to Rows and Columns for Tableau to do analysis
Standard Solutions for Web Services Data

JSON Connector

Only connects to files on a file system

Web Data Connector

Creates an Extract which only refreshes on a schedule
Can store one set of user credentials (that of the publisher)

Extract API

Can generate an Extract on demand if incorporated into its own service
Can append to existing Extract

Good for creating Extracts of large data sets which will remain relatively static and long-lived

"Locking Issue": If multiple users request data at the same time, must queue those requests so that they are all added to the updated Extract before it is published
Using a Relational Database to Store Web Service Responses

Tableau requires data be in a relational format

This is why Web Data Connector, JSON Connector, and Extract API create Extracts

The majority of the work is retrieving the web service responses and mapping it to a table

Relational Databases are very good at inserting large numbers of records very quickly

Solution: Store the web service responses in an RDBMS table and connect Tableau to that table
RDBMS Storing Web Service Responses: Embedded Application

1. Send Authentication Info (Token, Attributes, etc).
2. Send Request for Data including Auth Info
3. Retrieve Object Response
4. Write tabular data to Response Table with Response ID in every row of a Column
5. Retrieve Response ID
6. Load Tableau Viz using JS API, with Response ID Parameter set to Response ID
7. Live Connection to Response Table, Data Source Filter on [Response ID Column] = [Response ID Parameter]

Your Code: Retrieve, Transform, Insert
Filtering on Response ID in Tableau

Response should be a long unique randomly generated string

INDEX the columns storing the Response ID in the DB

The Tableau Parameter name to receive this value can be short
p1, s1, etc.

Create a Calculated Field to filter to the Parameter value
[Response ID] = [p1]

Create a Data Source Filter on the Calculated Field
Assumptions for Storing Web Service Responses in Database Table

Web Service responses must be a reasonable size and return fast.

Each Web Service Endpoint / Response type will require its own Table to be defined in the Relational Database.

Tableau workbooks are Embedded in an Application which handles the web service and the database insertions.
What about non-embedded use cases?

So far we’ve considered situations where there is an external application able to call this extra code to load the Web Services response into a Database.

The Tableau Extensions API allows integrating additional web-based components into a standard Tableau Dashboard.

The code which does the Request->Convert->Load process moves into the Extension.
RDBMS Storing Web Service Responses: Extension in Dashboard

1. Send Response ID Info (Token, Attributes, etc).
2. Send Request for Data including Auth Info
3. Retrieve Object Response
4. Write tabular data to Response Table with Response ID in every row of a Column
5. Retrieve Response ID
6. Load Tableau Viz using JS API, with Response ID Parameter set to Response ID
7. Live Connection to Response Table, Data Source Filter on [Response ID Column] = [Response ID Parameter]

Your Code: Retrieve, Transform, Insert

Tableau Workbook

Tableau Extension

Data Web Service

SSO / Auth

Response DB
What have we Covered?
What we done learned you

Row Level Security: What it is

Standard Row Level Security in Tableau: How it do

Advanced Row Level Security Solutions
  Securely Filtering at Load Time: Check yourself before you wreck yourself
  XML Modification of Template Files: Hot sauce for multi-tenant deployment
  “Live” Connecting to Web Service Data Sources: Serve up your microservices
Please complete the session survey from the Session Details screen in your TC18 app.
Resources
Standard Row Level Security & Parameters

https://tableauandbehold.com/row-level-security/

How to set up your Database for Row Level Security in Tableau

Securely Passing Parameters into a Tableau Viz at Load Time

Using Pass-Through Functions (RAWSQL) for Row-Level Security

Defusing Row Level Security in Your Extracts (Before They Blow Up) Part Two
XML Template Publishing

https://github.com/bryantbhowell/tableau_tools

https://github.com/tableau/document-api-python

https://interworks.com/work/analytics/power-tools-for-tableau/

The Tenets of Tableau Templates on Multi-tenants
Questions?