React-D3

Developer And User Manual

Author:
Simon Thelin

Supervisor:

July 11, 2017

Contents

1 Introduction

This developer and user manual will explain how to configure the environments and how to understand the code delivered with this manual. Follow every step of this manual and everything will be up running as it should.

2 Data-set

The data-set is a csv file that contains a number of attributes. The file is attached with this manual and it is called logons.csv.

3 Desired outcome

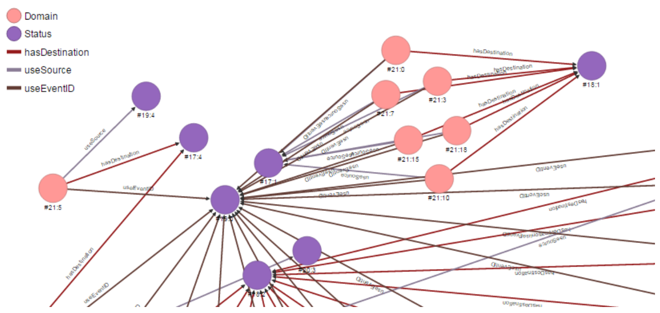
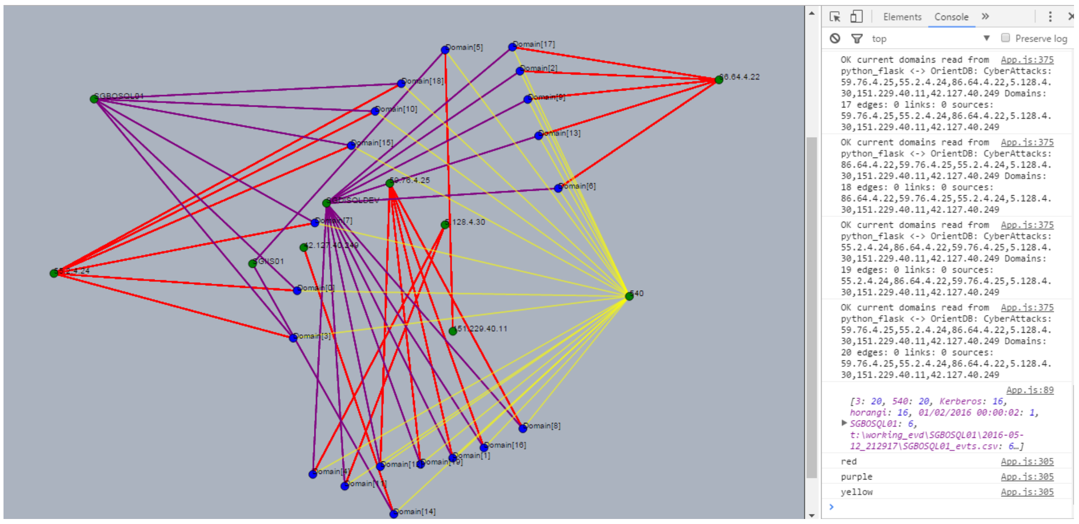


Figure 1: *OrientDB*



```
OK current domains read from  App.js:375
python_flask <-> OrientDB: CyberAttacks:
59.76.4.25,55.2.4.24,86.64.4.22,5.128.4.
30,151.229.40.11,42.127.40.249 Domains:
17 edges: 0 links: 0 sources:
59.76.4.25,55.2.4.24,86.64.4.22,5.128.4.
30,151.229.40.11,42.127.40.249
OK current domains read from  App.js:375
python_flask <-> OrientDB: CyberAttacks:
86.64.4.22,59.76.4.25,55.2.4.24,5.128.4.
30,151.229.40.11,42.127.40.249 Domains:
18 edges: 0 links: 0 sources:
86.64.4.22,59.76.4.25,55.2.4.24,5.128.4.
30,151.229.40.11,42.127.40.249
OK current domains read from  App.js:375
python_flask <-> OrientDB: CyberAttacks:
55.2.4.24,86.64.4.22,59.76.4.25,5.128.4.
30,151.229.40.11,42.127.40.249 Domains:
19 edges: 0 links: 0 sources:
55.2.4.24,86.64.4.22,59.76.4.25,5.128.4.
30,151.229.40.11,42.127.40.249
OK current domains read from  App.js:375
python_flask <-> OrientDB: CyberAttacks:
59.76.4.25,55.2.4.24,86.64.4.22,5.128.4.
30,151.229.40.11,42.127.40.249 Domains:
20 edges: 0 links: 0 sources:
59.76.4.25,55.2.4.24,86.64.4.22,5.128.4.
30,151.229.40.11,42.127.40.249
                                   App.js:89
  [3: 20, 540: 20, Kerberos: 16,
  horangi: 16, 01/02/2016 00:00:02: 1,
▶ SGBOSQL01: 6,
  t:\working_evd\SGBOSQL01\2016-05-
  12_212917\SGBOSQL01_evts.csv: 6…]
red                               App.js:305
purple                            App.js:305
yellow                            App.js:305
>
```

Figure 2: *React-D3*

4 Setup orientdb

There are two methods available to install OrientDB with some variations on each depending on the given operating system. The first method is to download a binary package from OrientDB. The other method is to compile the package from the source code. Begin with downloading it from[1]. Unzip the folder and place it where it can be found easily. If there is further questions about the installation please inspect the OrientDB installation manual[4].

When the installation is done locate the folder and run following commands from the cmd:

```
cd bin
./server.sh
```

If it does not work try instead:

```
cd bin
./server.bat
```

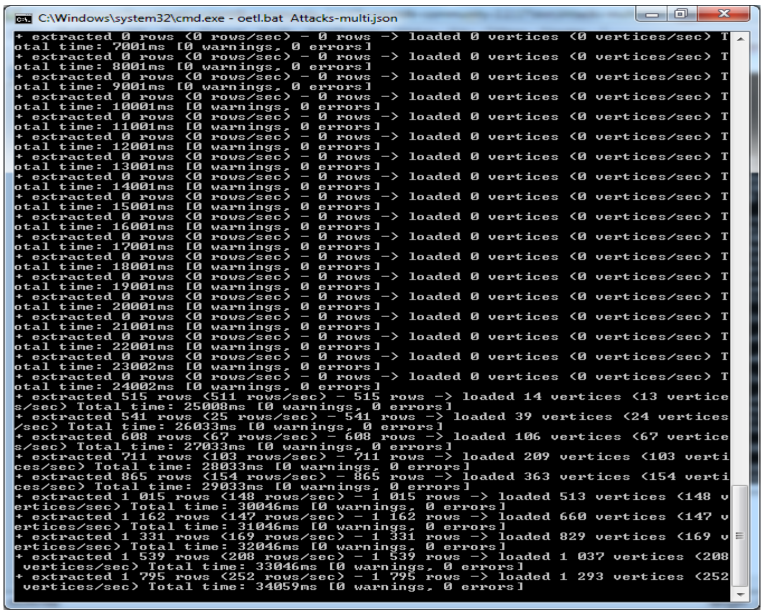
Once OrientDB is running, enter the following URL in a browser window: http://localhost:2480. This is the studio one of the most advanced Web tool for Databases.

Furthermore notice the video attached above. It provides the setup in a video to make sure there is no doubts.

4.1 Upload data

Now when the database is up running the process can continue. The data is going to be uploaded from the logons.csv file with using a json file. In this case called Attacks-multi.json[9]. It is located on the github provided with this manual.

Locate the bin folder orientdb-community-x.x.xx version and drag the file into the folder. Please remember to correct the paths within the Attacks-multi.json. The code is explained on the next page.



C:\Windows\system32\cmd.exe - oetl.bat Attacks-multi.json

```
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 7001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 8001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 9001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 10001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 11001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 12001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 13001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 14001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 15001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 16001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 17001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 18001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 19001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 20001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 21001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 22001ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 23002ms [0 warnings, 0 errors]
+ extracted 0 rows (0 rows/sec) - 0 rows -> loaded 0 vertices (0 vertices/sec) T
otal time: 24002ms [0 warnings, 0 errors]
+ extracted 515 rows (511 rows/sec) - 515 rows -> loaded 14 vertices (13 vertice
s/sec) Total time: 25008ms [0 warnings, 0 errors]
+ extracted 541 rows (25 rows/sec) - 541 rows -> loaded 39 vertices (24 vertices
/sec) Total time: 26033ms [0 warnings, 0 errors]
+ extracted 608 rows (67 rows/sec) - 608 rows -> loaded 106 vertices (67 vertice
s/sec) Total time: 27033ms [0 warnings, 0 errors]
+ extracted 711 rows (103 rows/sec) - 711 rows -> loaded 209 vertices (103 verti
ces/sec) Total time: 28033ms [0 warnings, 0 errors]
+ extracted 865 rows (154 rows/sec) - 865 rows -> loaded 363 vertices (154 verti
ces/sec) Total time: 29033ms [0 warnings, 0 errors]
+ extracted 1 015 rows (148 rows/sec) - 1 015 rows -> loaded 513 vertices (148 v
ertices/sec) Total time: 30046ms [0 warnings, 0 errors]
+ extracted 1 162 rows (147 rows/sec) - 1 162 rows -> loaded 660 vertices (147 v
ertices/sec) Total time: 31046ms [0 warnings, 0 errors]
+ extracted 1 331 rows (169 rows/sec) - 1 331 rows -> loaded 829 vertices (169 v
ertices/sec) Total time: 32046ms [0 warnings, 0 errors]
+ extracted 1 539 rows (208 rows/sec) - 1 539 rows -> loaded 1 037 vertices (208
 vertices/sec) Total time: 33046ms [0 warnings, 0 errors]
+ extracted 1 795 rows (252 rows/sec) - 1 795 rows -> loaded 1 293 vertices (252
 vertices/sec) Total time: 34059ms [0 warnings, 0 errors]
```

Figure 3: *Uploading data-set*

Figure 3 show the process when the following commands are entered in the cmd:

```
cd bin
./oetl.sh Attacks-multi.json
```

If it does not work try instead:

```
cd bin
./oetl.bat Attacks-multi.json
```

The database CyberAttacks is now created with all the vertices and edges it needs to be able to create the relational graph that is going to be displayed. The acutal script is explained below:

```
{
  "config": {
  },
  "source" : {"file": { "path": "C:/"path"/logons.csv" }},
  "extractor" : {
    "csv": {}
  },
```

First the path of the data-set logons.csv need to be added and the specific extractor is defined as the csv format.

```
  "transformers" : [
        { "vertex":{"class":"Status"}},
        { "vertex":{"class":"Domain" }},
        { "edge":{"class":"hasDomain", "joinFieldName": "Domain", "lookup": "Status.Domain", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"hasNetwork", "joinFieldName": "Type", "lookup": "Status.Type", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"useEventID", "joinFieldName": "EventID", "lookup": "Status.EventID", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"hasUser", "joinFieldName": "User", "lookup": "Status.User", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"useSource", "joinFieldName": "Source", "lookup": "Status.Source", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"hasDestination", "joinFieldName": "Destination", "lookup": "Status.Destination", "unresolvedLinkAction": "CREATE"}},
        { "edge":{"class":"useLogFile", "
```

```
           joinFieldName": "LogFile", "lookup": "
           Status.LogFile", "unresolvedLinkAction"
           : "CREATE"}},
  ],
```

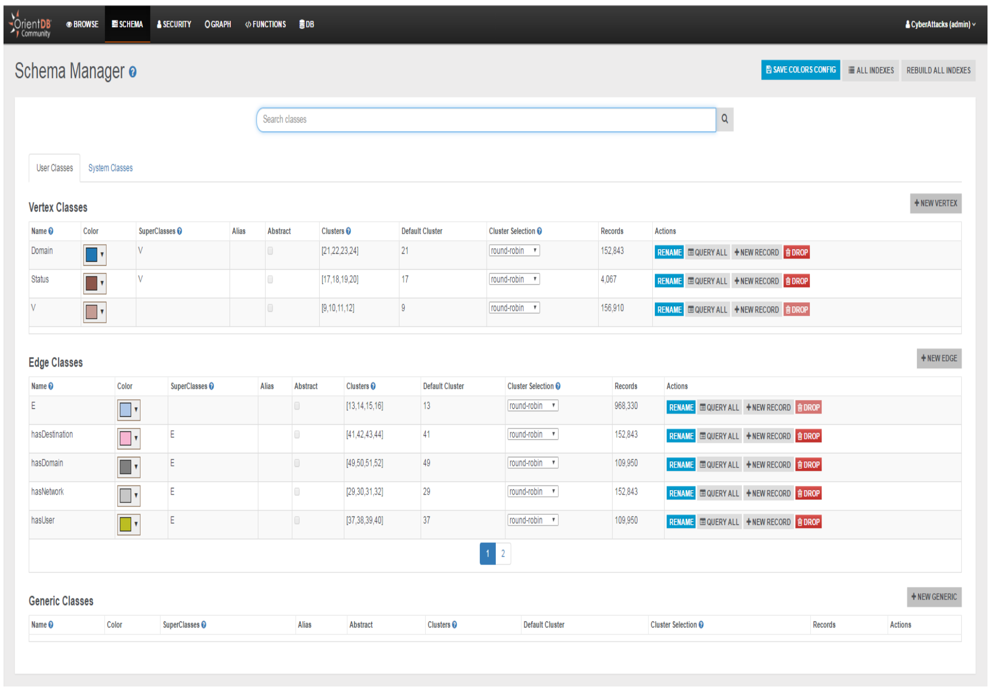
Then all the vertices and edges are created. The edges also has certain values they will search for. For an example useSource will join the Source column value and connect itself to that.

```
  "loader" : { "orientdb": {
      "dbURL": "plocal:C:/"path"/orientdb-
         community-x.x.xx/databases/CyberAttacks",
      "dbUser": "admin",
      "dbPassword": "admin",
      "dbAutoDropIfExists": true,
      "dbAutoCreate": true,
      "standardElementConstraints": false,
      "tx": false,
      "wal": false,
      "batchCommit": 1000,
      "dbType": "graph",
      "classes":[
          {"name": "Status", "extends":"V"},
          {"name": "Domain", "extends":"V"},
          {"name": "useDomain", "extends":"E"},
          {"name": "hasNetwork", "extends":"E"},
          {"name": "useEventID", "extends":"E"},
          {"name": "hasUser", "extends":"E"},
          {"name": "hasDestination", "extends":"E
             "},
          {"name": "useLogFile", "extends":"E"},
          ], "indexes":[
          {"class":"Status", "fields":["Domain:
             string", "Type:string", "AuthProtocol
             :string", "LogonType:integer", "
             Destination:string",
          "Source:string", "User:string", "
             DateTime:string",
          "LogFile:string", "EventID:integer"], "
             type":"NOTUNIQUE_HASH_INDEX" },
          {"class":"Domain", "fields":["Domain:
             string"], "type":"
             NOTUNIQUE_HASH_INDEX" }
```

```
                ]
        }
    }
}
```

Then it will setup the loader which is OrientDB and set the path for the database. Then the user-name and password is set for the database in this case just admin for user-name and admin for password. Then if the database already exists. Lets say there is changes that the developer want to do on the database. Then it is efficient to drop the existing database. This script will handle this automatically. It is a fact that the logons.csv contains a lot of data. Therefore it must be considered to make the uploading rate faster which will improve our development cycle. This is done with setting the tx to false and wal to false and to add a batchCommit to 1000. The type in this case is a graph because that is what the application want to work with. The classes are defined just as the ones that was created before. Then the indexes are manipulated so that the status vertex can hold all needed attributes. Both vertexes are allowed to add duplicates of the same data because in this case the same event-id can occur several times.

Now enter the studio environment and it should look like this:



Vertex Classes

Name	Color	SuperClasses	Alias	Abstract	Clusters	Default Cluster	Cluster Selection	Records	Actions
Domain	🟦	V		☐	[21,22,23,24]	21	round-robin	152,843	RENAME QUERY ALL NEW RECORD DROP
Status	🟫	V		☐	[17,18,19,20]	17	round-robin	4,067	RENAME QUERY ALL NEW RECORD DROP
V	🟫			☐	[9,10,11,12]	9	round-robin	156,910	RENAME QUERY ALL NEW RECORD DROP

Edge Classes

Name	Color	SuperClasses	Alias	Abstract	Clusters	Default Cluster	Cluster Selection	Records	Actions
E	🟦			☐	[13,14,15,16]	13	round-robin	968,330	RENAME QUERY ALL NEW RECORD DROP
hasDestination	🟪	E		☐	[41,42,43,44]	41	round-robin	152,843	RENAME QUERY ALL NEW RECORD DROP
hasDomain	⬛	E		☐	[49,50,51,52]	49	round-robin	109,950	RENAME QUERY ALL NEW RECORD DROP
hasNetwork	⬜	E		☐	[29,30,31,32]	29	round-robin	152,843	RENAME QUERY ALL NEW RECORD DROP
hasUser	🟨	E		☐	[37,38,39,40]	37	round-robin	109,950	RENAME QUERY ALL NEW RECORD DROP

1 2

Generic Classes

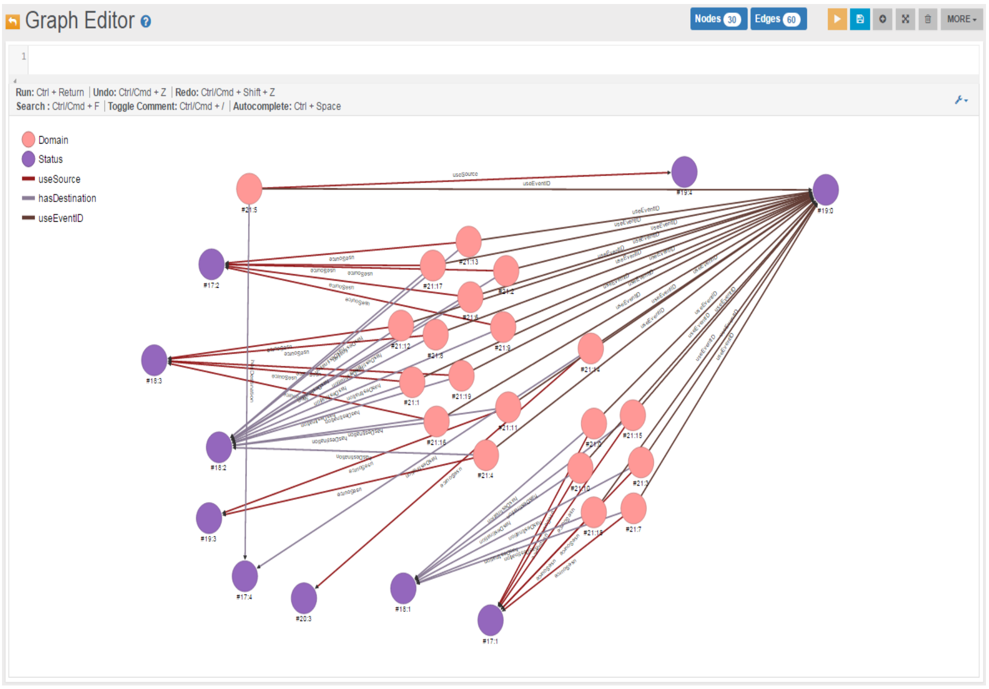
Name	Color	SuperClasses	Alias	Abstract	Clusters	Default Cluster	Cluster Selection	Records	Actions

4.1.1 OrientDB user input

Enter the graph tab inside the studio environment and enter following commands:

- SELECT * FROM useSource WHERE @rid > "#-1:-1" ORDER BY @rid ASC LIMIT 20
- SELECT * FROM hasDestination WHERE @rid > "#-1:-1" ORDER BY @rid ASC LIMIT 20
- SELECT * FROM useEventID WHERE @rid > "#-1:-1" ORDER BY @rid ASC LIMIT 20

Enter each of these commands and run each one with ctrl+enter. The output will be:



The possibility to choose much greater values to search for is available. But in this case 20 domains are called upon to make it easier to understand.

5 Setup Python

First of all check if Python is already installed by entering python in a command line window. The response from a Python interpreter it will include a version number in its initial display. Generally any recent version will do because as Python makes every attempt to maintain backwards compatibility.

Furthermore if Python needs to be installed. Download the most recent version. This is the one with the highest number that isn't marked as an alpha or beta release. Please see the Python downloads page[5] for the most up to date versions of Python 2 and Python 3. They are available via the yellow download buttons on that page.

For advice on choosing between Python 2 and Python 3 see Python 2 or 3.

For Windows: the most stable Windows downloads are available from the Python for Windows page.

For Mac see the Python for Mac OS X page[6]. MacOS 10.2 (Jaguar), 10.3 (Panther), 10.4 (Tiger) and 10.5 (Leopard).

For Red Hat, install the python2 and python2-devel packages.

For Debian or Ubuntu, install the python2.x and python2.x-dev packages.

For Gentoo, install the '=python-2.x*' ebuild (It may have to be unmasked first).

For other systems the installation can be made from another source. Furthermore see the general download page[5].

6 Setup Flask Project

6.1 PyCharm

When Python is installed on the given computer the procedure can move on to installing PyCharm. Begin with downloading the latest version of PyCharm from the JetBrains website[7]. There are the versions for Windows, OS X and Linux. Depending on your operating system. Please follow the jetbrains manual depending on what system is used[2].

6.2 Flask project

The source code can be found for downloading[8]. To create a new project from existing source code do following:

- On the main menu choose File | Open.
- In the dialog that opens, select the directory that contains the desired source code.

  Note that applications created externally are marked with the regular directory icon /help/img/idea/2016.3/folder.png.
- Click OK.

  Specify whether you want the new project to be opened in a separate window or close the current

However if the developer want to create the flask project from the beginning this can also be done. To create a Flask project please follow these steps:

- On the main menu, choose File | New | Project, or click the New Project button in the Welcome screen. Create New Project dialog box opens.
- In the Create New Project dialog box, specify the following:

  Project name and location

  Project type Flask project.
- In the Python Interpreter drop-down list. Select the Python SDK that is going to be used. If the desired interpreter is not found in the list, click and choose the interpreter type. Refer to the section Configuring Available Python Interpreters. If Flask is missing in the selected interpreter will display an information message that Flask will be downloaded.

- Click (More Settings), and specify the following:

  From the drop-down list, select the template language to be used. The directory where the templates will be stored.

- Click Create.

PyCharm creates an application and produces specific directory structure which can be explored in the Project tool window. Furthermore PyCharm creates a stub Python script with the name <project name>.py. Which provides a simple "Hello, World!" example.

6.3 Code implementation Flask server

```
# Author: Simon Thelin
# version: 1.2
# date: 2017-03-10
# -*- coding: utf-8 -*-
import pyorient
from flask import Flask, render_template, jsonify
#Allow cross-origin resource sharing (CORS)
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
```

First of all the needed libraries are imported. PyCharm will help with the installing of all needed libraries just press alt+enter and it will be done. It is also possible to install in the console. Then just type 'pip install "library"' and it works fine as well. In this case pyorient is used. OrientDB supports all JVM languages for server-side scripting. Using the PyOrient module. Development with database applications for OrientDB using the Python language is enabled. In this application cross -origin resource(CORS) must be activated. When enabling CORS there is a desire to enable it for all use cases on a domain. Therefore no mucking around with different allowed headers or methods and so on. Furthermore the default state submission of cookies across domains is disabled due to the security implications. Then an instance of this class is made. The first argument is the name of the applications module or package. If a single module (as in this example) ___name___ should be used because depending on if it is started as application or imported as module the name will be different ('___main___' versus the actual import name). This is needed so that Flask knows where to look for templates or static files. The library jsonify is imported so that the parsing to JSON format is available when the React-D3 application fetch the data. The render_template is there to render the html file contained within the project. But this is never used in our application.

```
data_to_json = []

client = pyorient.OrientDB( "localhost", 2424 )
client.set_session_token( True )
client.connect( "root_user", "rute_pass" )
client.db_open( "CyberAttacks", "admin", "admin")

loop = (client.command( "select * from Domain
   ORDER BY @rid ASC LIMIT 20" ))
```

An array is created that will contain each domain object. In this case of JSON format. PyOrient is composed of two layers. At its foundation is the python wrapper around the OrientDB binary protocol. Furthermore OrientDB has its own SQL language. It is the Object-Graph Mapper (or OGM). The OGM layer is documented separately. An init is made to the client and then the session token is enabled. Furthermore in this use case there is a need to maintain a client connection across several sessions. For instance, in a web application there might be a will to set an identifier for a shopping cart or use sessions to maintain a local history of the users interactions with the site. Furthermore a connection is made and proceeds to the given database of CyberAttacks. Then the a call to the Domain vertex is made. This is done on th basis of 20 domains. Therefore not this can be changed to whatever reasonable number up to the max value of the number of rows available.

```
for result in loop:
    data_to_json.append({'Domain': result.Domain,
        'AuthProtocol': result.AuthProtocol, '
       LogonType':result.LogonType, 'Destination':
        result.Destination,'Source': result.Source
       ,'User': result.User,'DateTime': result.
       DateTime, 'LogFile': result.LogFile, 'Type'
       : result.Type, 'EventID': result.EventID})

@app.route("/getData")
def index():
    return jsonify(data_to_json)

if __name__ == "__main__":
    app.run()
```

Then the result is going through a loop and add every value to each attribut. Null is also an acceptable value and will be set if the value is missing. Then the route() decorator is used to tell Flask what URL should trigger the function. The React-D3 code will fetch from this and note the jsonify of the data-set. And the application will run. Note no debugging features activated and it should not be activated in this case.

6.3.1 User input

Worth mentioning is this line of code:

```
loop = (client.command( "select * from Domain
   ORDER BY @rid ASC LIMIT 20" ))
```

If another number of domains want to be reached then this need to be changed to a greater number such as 100 or 10000.

7 Setup React environment

7.1 WebStorm

Begin with downloading the latest version of WebStorm from the Jet-Brains website[10] There are the versions for Windows, OS X and Linux. Depending on your operating system:

- Windows: Run the .exe file and follow the instructions of WebStorm Setup wizard.
- OS X: Open the .dmg package, and drag WebStorm to the Applications folder.
- Linux: Unpack the .tar.gz archive into any directory within your home directory.

And run the procedure of the installation that Jetbrains provide.

Make sure to look trough the quick-start manual Jetbrains provide if further questions arise [3].

7.2 Setup React Project

WebStorm recognizes JSX code and provides syntax highlighting for the code completion and navigation. Furthermore also code analysis for it. Code completion and navigation is also provided for ReactJS methods.

WebStorm can also provide code completion for HTML tags and component names that has been defined inside methods in JavaScript or inside other components.

Completion also works for imported components with ES6 style syntax. It is possible to navigate from a component name to its definition with Ctrl+B or see a definition in a popup with Ctrl+Shift+I.

In JSX tags WebStorm can provide coding assistance for ReactJS-specific attributes such as className or classID. More exactly class names that can be auto-completed classes defined in the projects CSS files.

Download the project from the github source[11]. Then follow these steps:

- On the main menu choose File | Open.
- In the dialog that will occur. Select the directory that contains the desired source code.

  Note that applications created externally are marked with the regular directory icon folder.png.
- Click OK and specify whether the project will be opened in a separate window or close the current project and reuse the existing one.

NodeJS has already installed all the needed libraries for the project and it is integrated in the console of WebStorm. Therefore the developer does not have to do this at the initial stage. However if the developer wants to add new features they need to be installed through the npm. Follow these steps if that is the case:

- Press alt+F12
- Enter npm install "library"
- The result will end up in the console if it was a success or not

7.3 Code implementation React-D3

Now when the project is up running lets make sure to analyze the actual code. However the whole code is not going to be covered in this manual because it would be to much to analyze. The focus will be on the most important features. Such as fetching the data. How is it stored within the application. And how is the actual connection of nodes and edges made. Lets start by looking at the fetching of the data.

```
    constructor(props) {
        super(props);
        this.state = {
            dataObjects: [],
            dataFetched: false
        }
    }
    /*
     *Handle the GET API request
     */
    handleGetData() {
        document.getElementById("button").
           disabled=true;
        axios.get('http://localhost:5000/getData
           ')
            .then((response) => this.setState({
                dataObjects: response.data,
                dataFetched: true
            }))
            .catch(function (error) {
                console.log(error);
            });
    }
```

The constructor will hold the dataObjects which in this case is the domains. There will also be an flag indicating if the fetching as been successful or not. The http request is made from the flask server and the data is retrieved in to our application.

```
var domainCounter = 0, edgeCounter = 0,
   linkCounter = 0;
var sources = [], destinations = [], eventID = []
   , result = [], domains = [];
/*
 * Will hold the information about each domain!
 */
```

```
var  data = {
    "nodes": [],
    "links": []
};
```

The actual data that will contain all the nodes and edges are initialized. With nodes and links. Where the nodes will be both the vertex and the edge where the links connect the nodes with the edges. We have also declared several variables to maintain the structure. Will be covered later on.

```
    this.state.dataObjects.map((objects) => {
        document.getElementById("button").
           disabled=false;
        data.nodes.push({
            "id": domainCounter,
            "AuthProtocol":objects.AuthProtocol,
            "Domain": objects.Domain,
            "DateTime": objects.DateTime,
            "text": "Domain[" + domainCounter + "
               ]",
            "Destination": objects.Destination,
            "EventID": objects.EventID,
            "LogFile": objects.LogFile,
            "LogonType": objects.LogonType,
            "Source": objects.Source,
            "Type": objects.Type,
            "User": objects.User,
            "EdgeID": -1,
            "color": "blue"
    });
    domainCounter++;
```

Here every domain node will be read and saved in the data-set of nodes. Notice that they are marked to have an edge id with -1. This will help the connection later on. The domainCounter will also increment for every added domain.

```
sources.push(objects.Source);
sources = this.sortReplicas(sources);
destinations.push(objects.Destination);
destinations = this.sortReplicas(destinations);
eventID.push(objects.EventID);
eventID = this.sortReplicas(eventID);
```

The application will add the source, destination and event-id values to arrays where they will be sorted. This to make sure there is no replicas within the graphical representation. This is also used within the actual algorithm when creating the nodes and edges with links.

```
    sortReplicas(array) {
        if(array!=null) {
            var units = array.map((name) => {
                return {count: 1, name: name}
            }).reduce((a, b) => {
                a[b.name] = (a[b.name] || 0) + b.
                    count
                return a
            }, {})
            array = Object.keys(units).sort((a, b
                ) => units[a] < units[b])
        }
        return array;
    }
```

The sorting is straight forward.

```
<button className="button" id='button' onClick={
    () => this.start()}>Render</button>
```

Then when the user wants to render the data-sets to creat the relational graph. A method called start will be called.

```
    start() {
        this.countOccurrence();
        console.log(result);
        this.createEdgeAndLinks(sources);
        this.createEdgeAndLinks(destinations);
        this.createEdgeAndLinks(eventID);
        flag=false;
        this.componentDidMount();
        document.getElementById("button").
            disabled=true;
    }
```

The first method call is countOccurrence. It will simply calculate all the number of a repeating attribute. This is for now mainly for debugging purpose.

```
    countOccurrence() {
        if(domains!=null) {
            for (var i = 0; i < domains.length;
                ++i) {
                if (!result[domains[i]]) {
                    result[domains[i]] = 0;
                }
                ++result[domains[i]];
            }
        }
    }
```

A straight forward solution. In the same moment as this happens the flag is set to false. Lets go back and look at the calls for createEdgeAndLinks.

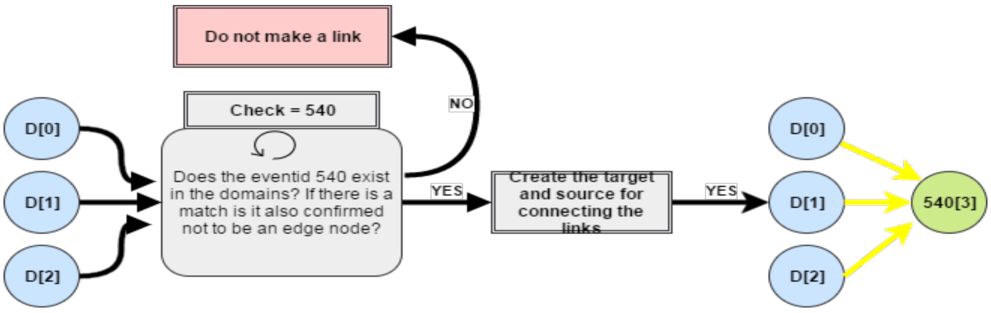
```
    createEdgeAndLinks(array) {
        for (var i = 0; i < array.length; i++) {
            data.nodes.push({
                "EdgeID": edgeCounter,
                "Source": array[i],
                "text": array[i],
                "color": "green"
            });
            edgeCounter++;
        }
        var source = [];
        var target = 0;

        for(var j=0; j<array.length; j++) {
            this.addLink(source, target, array[j]
                , array);
            source = [];
            target = 0;
        }
    }
```

This will now create the edges for the application. So lets say there is 20 domains where all of them have the event-id 540. But there is no need to have 20 edges with the same value. So based upon the replica check there will now only be one edge created. Then it will also make a call each time to the existing nodes which in this case are the domains. Notice the edgeCounter increments for every edge. Lets take a look inside addLink.

```
   addLink(source, target, check, array) {
        var color = "";
        for(var k=0; k<array.length; k++) {
            for (var j = 0; j < (domainCounter+edgeCounter); j++) {
                /*
                 * Find the eventID from the domain!
                 */
                if (parseInt(data.nodes[j].EventID, 10) === parseInt(check, 10) && data.nodes[j].EdgeID === -1) {
                    source.push(j);
                    color = "yellow";
                }
                /*
                 * Find the target from the edge!
                 */
                if (parseInt(data.nodes[j].EventID, 10) === check && data.nodes[j].EdgeID >= 0) {
                    target = j;
                }
            }
            /*
             * Add the link!
             */
            console.log(color);
            for (var p = 0; p < source.length; p++) {
                data.links.push({
                    "source": source[p],
                    "target": target,
                    "value": source + "-" + target,
                    "color": color
                });
                linkCounter++;
            }
        }
    }
```

This is the current solution. The links will be created and different colors to the links will be chosen depending on what attribute it contains. The code above is confined to the event-id. The same procedure is made for the other attributes source and destinations. The algorithm make sure to find the actual edges which are indicated with the value of -1. An illustration is made below:


Do not make a link
Check = 540
NO
Does the eventid 540 exist in the domains? If there is a match is it also confirmed not to be an edge node?
YES
Create the target and source for connecting the links
YES
D[0]
D[1]
D[2]
D[0]
D[1]
D[2]
540[3]

When looking back at the call to the start() method. There is also a flag that is set to false. This will make the componentDidMount activate.

```
componentDidMount() {
        if(!flag) {
            const {width, height} = this.props;

            simulation = d3.forceSimulation(data.
              nodes)
                .force("links", d3.forceLink(data
                  .links).distance(700))
                .force("charge", d3.forceManyBody
                  ().strength(-120))
                .force('center', d3.forceCenter(
                  width / 2, height / 2));

            const svg = d3.select(this.refs.
              mountPoint)
                .append("svg")
                .attr("width", width)
                .attr("height", height);

            link = svg.selectAll('line')
                .data(data.links)
                .enter()
                .append('line')
                .style('stroke-width', 3.5)
                .style('stroke', function (d) {
                    return d.color
                })
                .style('stroke-opacity', 0.6);

            node = svg.selectAll("circle")
                .data(data.nodes)
                .enter()
                .append("circle")
                .attr("r", 10)
                .style('stroke', '#000000')
                .style('stroke-width', 1.5)
                .style("fill", function (d) {
                    return d.color
                })
                .call(d3.drag()
                    .on("start", this.dragstarted)
                    .on("drag", this.dragged)
                    .on("end", this.dragended));

           label = svg.selectAll("text")
               .data(data.nodes)
               .enter()
               .append("text")
               .text(function (d) { return d.text; })
               .style("text-anchor", "left")
               .style("fill", "#060005")
               .style("font-family", "Arial")
               .style("font-size", 20);

           this.tick(link, node, label);

        }
        if(flag) {
            flag=false;
            this.handleGetData();
        }
    }
}
```

This will make sure that the nodes and edges and links are displayed on the react page in a proper way. The solution is very straight forward. It will also make a call to the tick function which will make sure that the nodes location are updated:

```
/**
* When the user want to drag the nodes
* restart the tick()
*/
restart_tick() {
    if(!flag) {
        this.tick(link, node, label);
    }
}
/**
* function tick that make sure to update the
* position of the nodes
*
* @param link
* @param node
* @param label
*/
tick(link, node, label) {
    d3.forceSimulation().on('tick', () => {
        link
            .attr("x1", function (d) {
                return d.source.x;
        })
        .attr("y1", function (d) {
                return d.source.y;
        })
        .attr("x2", function (d) {
                return d.target.x;
        })
        .attr("y2", function (d) {
                return d.target.y;
        });
        node
            .attr("cx", function (d) {
                return d.x;
            })
            .attr("cy", function (d) {
                return d.y;
            });
        label
            .attr("x", function (d) {
                return d.x;
            })
            .attr("y", function (d) {
                    return d.y;
            });
    });
}
```

Worth mentioning is the svg element. Inside the svg element all the nodes and edges and links are confined. This svg element is what gets rendered in at the react page mentioned before. D3 is the library that is being used. The current solution for drag() events is not optimal but it works. It receives the oldest position every time the user clicks the svg element if the tick() is out of time.

```
/**
* Function for drag functionality
* @param d
*/
dragstarted(d) {
    if (!d3.event.active)
        simulation.alphaTarget(0.3).restart();
    d.fx = d.x;
    d.fy = d.y;
}
/**
* Function for drag functionality
* @param d
*/
dragged(d) {
    d.fx = d3.event.x;
    d.fy = d3.event.y;
}
/**
* Function for drag functionality
* @param d
*/
dragended(d) {
    if (!d3.event.active)
        simulation.alphaTarget(0);
    d.fx = null;
    d.fy = null;
}
```

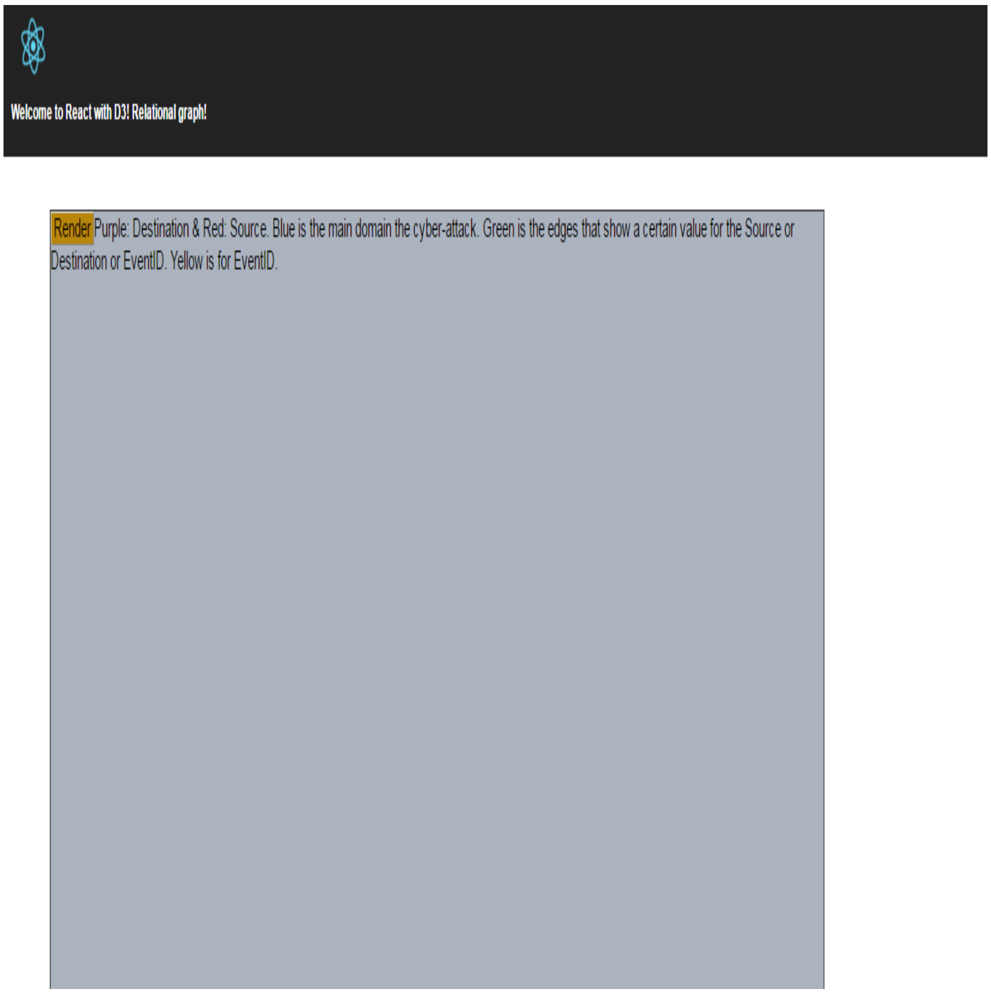
They make it possible to move around the nodes during the initialization time and after the initialization. This can be furthered analyzed in the code itself.

7.3.1 User input

To run this project enter in the console by pressing alt+F12. then enter:

- npm install
- npm start

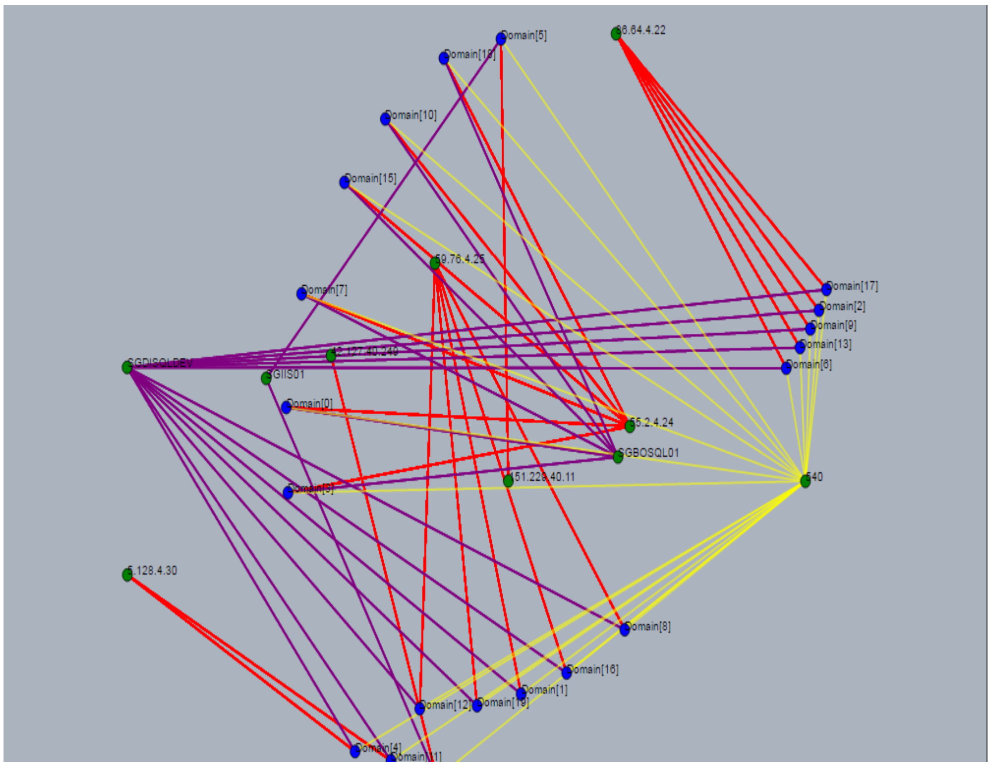
The application will later on be available at http://localhost:3000/. Enter this in the web-browser and the result will look like:



Welcome to React with D3! Relational graph!

Render Purple: Destination & Red: Source. Blue is the main domain the cyber-attack. Green is the edges that show a certain value for the Source or Destination or EventID. Yellow is for EventID.

Press the render button and the result will be displayed. Feel free to move the nodes around.



Happy analyzing.

8 References

[1] OrientDB Community Edition. `http://orientdb.com/download/`. Accessed: 2017-03-17.

[2] Installation PyCharm Details. `https://www.jetbrains.com/pycharm-edu/quickstart/installation.html#macos`. Accessed: 2017-03-17.

[3] Quickstart WebStorm. `https://www.jetbrains.com/help/webstorm/2016.3/quick-start-guide.html`. Accessed: 2017-03-17.

[4] OrientDB Installation Manual. `http://orientdb.com/docs/last/Tutorial-Installation.html`. Accessed: 2017-03-17.

[5] Install Python. `https://www.python.org/downloads/`. Accessed: 2017-03-17.

[6] Install Python MAC. `https://www.python.org/downloads/mac-osx/`. Accessed: 2017-03-17.

[7] Install PyCharm. `https://www.jetbrains.com/pycharm/download/#section=windows`. Accessed: 2017-03-17.

[8] Python Flask Github. `https://github.com/Thelin90/CyberAttacks_python-flask-server`. Accessed: 2017-03-17.

[9] OrientDB Github. `https://github.com/Thelin90/CyberAttacks_OrientDB`. Accessed: 2017-03-17.

[10] Installing WebStorm. `https://www.jetbrains.com/webstorm/download/#section=windows`. Accessed: 2017-03-17.

[11] React-D3 Github. `https://github.com/Thelin90/CyberAttacks`. Accessed: 2017-03-17.