CYBER SECURITY VISUALIZATION USING JAVASCRIPT D3 LIBRARY

Raoul Choussi Dzodoum (*Author*)
Electrical and Computer Engineering Department
Morgan State University
Baltimore, USA

Abstract- Within my study Computer/Electrical engineering, I am interested in programing and different simulation. It was a great opportunity for me work on a cyber-security project in which I had to process a csv file containing cyber security data and come up with suitable visualization to help analyst understand the data. While working on this project during my last two semesters, several actions have contributed to achieve a number of goals. Since the project required a 3d output, I first started to be familiar with the JavaScript d3 library and then I learned html syntaxes. Finally I was able to start coding a come up with a 3d visualization.

I. INTRODUCTION

This report is a short description of my two semesters working on a project using JavaScript d3 library. The project was supervised by Dr. Kofi Nyarko; professor at Morgan State University. In this project, the work was concentrated on taking cyber security data, process it and come up with useful visualization to help the user understand the data.

II. PURPOSE

This report discusses the results of the work done in the conversion of a "csv file" data I was given into a 3d visualization. It also discusses the methodological approach developed and the integration of various tools used during the execution of this project. Finally, it discusses the results and the improvement that could be done. The main goals of the projects were:

- Be able to understand the "csv file" code that was given.
- Come up with a design that will help analyst understand the data.
- Explain the results and the method used

III. MATERIALS

1. Computer
2. Browser
3. Text editor

IV. TOOLS

JavaScript

JavaScript was appropriate for this project because:

- Easy to implement
- Supports 3d visualization

V. METHODOLOGY

To implement the above goals, the following methodology needs to be followed:

1. I started to be familiar codes on websites d3js.org
    - to understand how the 3D library works
2. Be familiar with html syntaxes
    - Very useful to write codes in JavaScript.
3. Examine the "csv file" that I was given
    - To determine how my output will look like.
4. Chose a sample design
    - to help the user understand the output easily
5. Choose a text editor.
    - To write my code
6. Start coding.

VI. CONTENTS OF THE CSV FILE

In order to determine how my output will look like, I had to understand the data in the file. The file was composed of data of systems that were conducting or experiencing cyber-attacks. Those data that were divided in two main categories: the source and the destination and each attack was associated with a specific "ID", the "time" of the attack, and an alert message when a specific attack was happening.

1- Source

It was composed of the of the country where the attack originated " src country," the name of the entity "src entity", and the source port number "src port."

2- Destination

It was composed of the of the country where the attack targeted "

Dst country," the name of the entity targeted "Dst entity", and the targeted port number "Dst port."

VII. CHOICE OF THE MAP

Since the data in the file was showing a relationship between sources and targets of different attacks, I decided pick an output where the analyst could see the connection between the different sources and targets. So my first map was an atomic model. However, I decide to change that model because although the connections were visible, the analyst was not able to determine which node was the source or the target. So I finally switch to the map called "directional forces layout diagram". The reason I used this map is because not only it shows the different connections between sources and target, the analyst can also determines which node is the source and which one is the target because arrows.

VIII. NOTES ON THE IMPLEMENTATION

1. VARIABLE DECLARATION

I declared the least number of variables in order to avoid the repetition of certain operations in the program. Most of the variables used in the program were strings. I then declared an empty object that will contains all the nodes that I created in the code.

2. THE INITIAL HTML SECTION

The first block of my code was the initial html section. In this part of the code I put all the libraries that I needed for the project.

```
<!DOCTYPE html>
<meta charset="utf-8">
<script type="text/javascript" src="d3/d3.v3.j
<style>
```

3. THE CASCADING STYLE SHEET

In this section of the code I loaded the cascading style sheet that creates the paths, the circles, and the text that I used in the code.

```
path.link {
  fill: none;
  stroke: #666;
  stroke-width: 1.5px;
}

circle {
  fill: #ccc;
  stroke: #fff;
  stroke-width: 1.5px;
}

text {
  fill: #000;
  font: 10px sans-serif;
  pointer-events: none;
}
```

Link.source : represent the different sources that are created for row of the csv file.

Link.target: represent the different targets that are created for row of the csv file.

4.LOAD THE "CSV FILE"

When I moved into JavaScript, the first part of my code loads the csv data file (force.csv) from our directory.

```
d3.csv("data/force.csv", function(error, li
```

5.FUNCTION THAT CONTAINS DATA FOR MY NODES.

```
links.forEach(function(link) {
    link.source = nodes[link.source] ||
        (nodes[link.source] = {name: link.source});
    link.target = nodes[link.target] ||
        (nodes[link.target] = {name: link.target});
    link.value = +link.value;
});
```

Links: represent the array of objects that was loaded from the csv file.

6.SOURCES AND TARGETS CHOICES

In order to make the choice of the different sources and target, I created a "drop down menu button" to make the different selections. The execution button loads the file into the program and then processes it.

```
  <b>Define Source</b>
  <select id="sourceChoose">
    <option value= "1">Source Entit
    <option value="2">Source Port</
    <option value="3">Source countr
  </select>

  <b>Define Destination</b>
  <select  id="destinationChoose">
    <option value= "4" >Destinatior
    <option value="5">Destination P
    <option value="6">Destination C
  </select>

  <button onclick="loadFile()">exec
</div>
```

IX. RESULTS AND DISCUSSION

As we can observe in the output we have a representation of the source countries and the destination countries from the csv file (figure 1). The directions of the arrows indicate which node is the country where the attack originated and which node is country targeted.in (figure 2) we have the output between the sources port and the destinations ports. Just like in the source country and destination country, here we have arrows indicated port were the attack originated and the country targeted. And from that we can make any combination between the different sources and destinations. I also created a mouse event where we can click on each node if we want to zoom it.

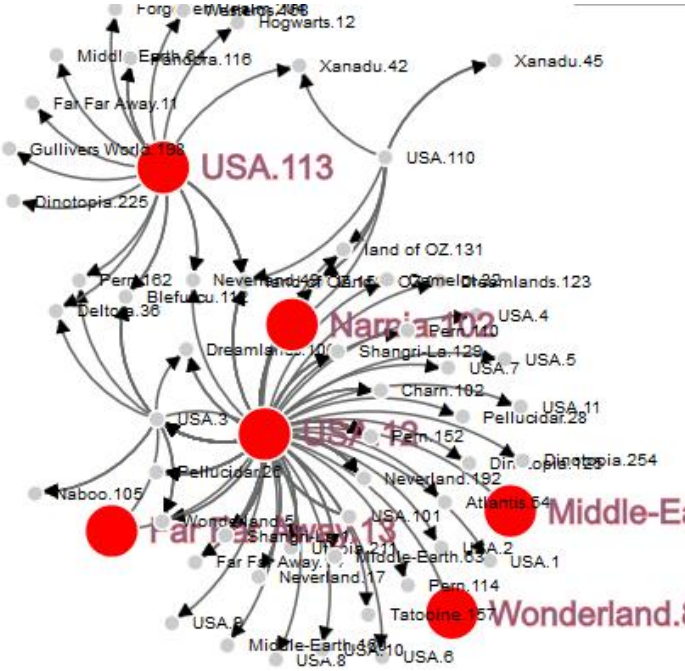
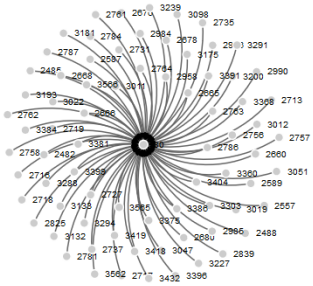


FIGURE 1



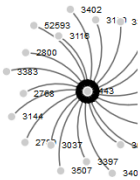


FIGURE 2

X. OBSACLE AND FUTURE WORK

The main problem in this project was to find the appropriate design to represent the data in 3d this really consume a lot of my time. For this project, I would like to create a mousse event that will display a message alert for each connection. Also, I would

like to create a legend that will represent the number of connections.

XI. CONCLUSION

In conclusion, the program runs without any errors. I was able to achieve the main goals of my project which was to come up with a useful representation of the data from the csv file to make the user understand it. We can clearly see the different connections between the different sources and destinations appear on the output. However, I would like to add some message alert for each connection and also create a legend for the different number of connections.

XII. ACKNOWLEDGMENT

I would like to thank Dr. Kofi Nyarko for assigning me this project and for his great help with the completion of this project. I would also like to thank some of my schoolmates for contributing to the accomplishment of this project.

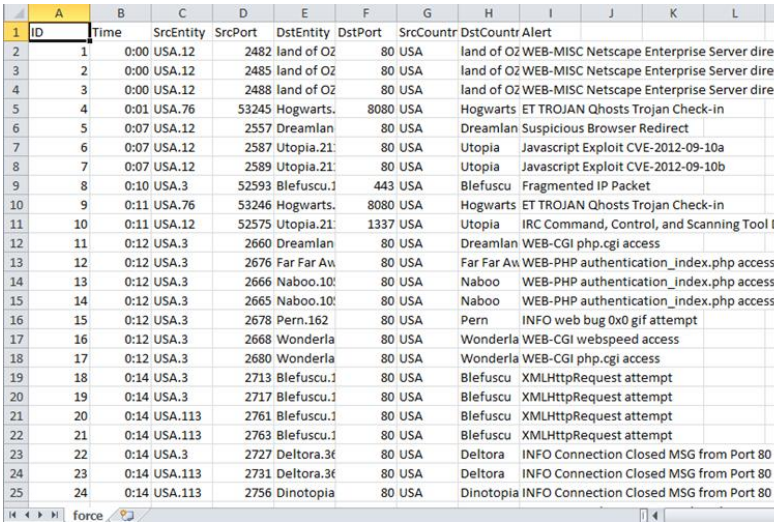
REFERENCES

https://github.com/mbostock/d3/wiki/Gallery

http://www.w3schools.com/

APPENDIX

I. CSV FILE



	A	B	C	D	E	F	G	H	I
1	ID	Time	SrcEntity	SrcPort	DstEntity	DstPort	SrcCountr	DstCountr	Alert
2	1	0:00	USA.12	2482	land of O2	80	USA	land of O2	WEB-MISC Netscape Enterprise Server dire
3	2	0:00	USA.12	2485	land of O2	80	USA	land of O2	WEB-MISC Netscape Enterprise Server dire
4	3	0:00	USA.12	2488	land of O2	80	USA	land of O2	WEB-MISC Netscape Enterprise Server dire
5	4	0:01	USA.76	53245	Hogwarts.	8080	USA	Hogwarts	ET TROJAN Qhosts Trojan Check-in
6	5	0:07	USA.12	2557	Dreamlan	80	USA	Dreamlan	Suspicious Browser Redirect
7	6	0:07	USA.12	2587	Utopia.21	80	USA	Utopia	Javascript Exploit CVE-2012-09-10a
8	7	0:07	USA.12	2589	Utopia.21	80	USA	Utopia	Javascript Exploit CVE-2012-09-10b
9	8	0:10	USA.3	52593	Blefuscu.1	443	USA	Blefuscu	Fragmented IP Packet
10	9	0:11	USA.76	53246	Hogwarts.	8080	USA	Hogwarts	ET TROJAN Qhosts Trojan Check-in
11	10	0:11	USA.12	52575	Utopia.21	1337	USA	Utopia	IRC Command, Control, and Scanning Tool
12	11	0:12	USA.3	2660	Dreamlan	80	USA	Dreamlan	WEB-CGI php.cgi access
13	12	0:12	USA.3	2676	Far Far Aw	80	USA	Far Far Aw	WEB-PHP authentication_index.php access
14	13	0:12	USA.3	2666	Naboo.10	80	USA	Naboo	WEB-PHP authentication_index.php access
15	14	0:12	USA.3	2665	Naboo.10	80	USA	Naboo	WEB-PHP authentication_index.php access
16	15	0:12	USA.3	2678	Pern.162	80	USA	Pern	INFO web bug 0x0 gif attempt
17	16	0:12	USA.3	2668	Wonderla	80	USA	Wonderla	WEB-CGI webspeed access
18	17	0:12	USA.3	2680	Wonderla	80	USA	Wonderla	WEB-CGI php.cgi access
19	18	0:14	USA.3	2713	Blefuscu.1	80	USA	Blefuscu	XMLHttpRequest attempt
20	19	0:14	USA.3	2717	Blefuscu.1	80	USA	Blefuscu	XMLHttpRequest attempt
21	20	0:14	USA.113	2761	Blefuscu.1	80	USA	Blefuscu	XMLHttpRequest attempt
22	21	0:14	USA.113	2763	Blefuscu.1	80	USA	Blefuscu	XMLHttpRequest attempt
23	22	0:14	USA.3	2727	Deltora.3(	80	USA	Deltora	INFO Connection Closed MSG from Port 80
24	23	0:14	USA.113	2731	Deltora.3(	80	USA	Deltora	INFO Connection Closed MSG from Port 80
25	24	0:14	USA.113	2756	Dinotopia	80	USA	Dinotopia	INFO Connection Closed MSG from Port 80

force

2. JavaScript code

```
  <!DOCTYPE html>
<meta charset="utf-8">
<script
src="http://d3js.org/d3.v3.js"></script>
<script type="text/javascript"
src="http://mbostock.github.com/d3/d3.js?
1.27.1"></script>

<style>

div.tooltip {
  position: absolute;
  text-align: center;
  width: 60px;
  height: 12px;
  padding: 8px;
  font: 10px sans-serif;
  background: #ddd;
  border: solid 1px #aaa;
  border-radius: 8px;
  pointer-events: none;
}

.link {
  fill: none;
  stroke: #666;
  stroke-width: 2px;
}

.link.colorRed{
  stroke: #FF0000;
}
```

```
.link.colorOrange{
  stroke: #FF7F00;
}

.link.colorYellow{
  stroke: #FFFF00;
}

.link.colorGreen{
  stroke: #00FF00;
}

.link.colorBlue {
  stroke: #0000FF;
}

.link.colorIndigo{
  stroke: #00FFFF;
}

.link.colorViolet{
  stroke: #FF00FF;
}

path.link {
  fill: none;
  stroke: #666;
  stroke-width: 1.5px;
}

circle {
  fill: #ccc;
  stroke: #fff;
  stroke-width: 1.5px;
}

text {
  fill: #000;
  font: 10px sans-serif;
  pointer-events: none;
}

ui-tooltip, .arrow:after {
    background: black;
    border: 2px solid white;
}
.ui-tooltip {
  padding: 10px 20px;
  color: white;
  border-radius: 20px;
  font: bold 14px "Helvetica Neue", Sans-
Serif;
  text-transform: uppercase;
  box-shadow: 0 0 7px black;
}
.arrow {
  width: 70px;
  height: 16px;
  overflow: hidden;
  position: absolute;
  left: 50%;
  margin-left: -35px;
  bottom: -16px;
}
.arrow.top {
  top: -16px;
  bottom: auto;
}
.arrow.left {
  left: 20%;
}
.arrow:after {
  content: "";
  position: absolute;
  left: 20px;
  top: -20px;
  width: 25px;
  height: 25px;
  box-shadow: 6px 5px 9px -9px black;
  -webkit-transform: rotate(45deg);
  -ms-transform: rotate(45deg);
  transform: rotate(45deg);
}
.arrow.top:after {
  bottom: -20px;
  top: auto;
}

</style>


<body>
<div style = "position: fixed;">

  <div class='space'>
    <b>Define Source</b>
    <select id="sourceChoose">
      <option value= "1">Source
Entity</option>
      <option value="2">Source
Port</option>
      <option value="3">Source
country</option>
    </select>

    <b>Define Destination</b>
    <select  id="destinationChoose">
      <option value= "4" >Destination
Entity</option>
      <option value="5">Destination
Port</option>
      <option value="6">Destination
Country</option>
    </select>
```

```
    <button
onclick="loadFile()">execute</button>
  </div>

  <div class='my-legend'>
    <div class='legend-title
space'>Legend: </div>
      <div class='legend-scale'>
        <ul class='legend-labels'>
          <li><span
style='background:#00FFFF;'></span><p
id="indigo"></p></li>
          <li><span
style='background:#0000FF;'></span><p
id="blue"></p></li>
          <li><span
style='background:#00FF00;'></span><p
id="green"></p></li>
          <li><span
style='background:#FFFF00;'></span><p
id="yellow"></p></li>
          <li><span
style='background:#FF7F00;'></span><p
id="orange"></p></li>
          <li><span
style='background:#FF0000;'></span><p
id="red"></p></li>
          <li><span
style='background:#FF00FF;'></span><p
id="violet"></p></li>
        </ul>
    </div>
  </div>
</div>

<script>


function loadFile(){
// get the data
d3.csv("force.csv", function(links) {

var nodes = {};

console.log(sourceChoose)

    var select1 =
document.getElementById("sourceChoose");
    var select2=
document.getElementById("destinationChoos
e");

    if (select1.value === "1" &&
select2.value === "4")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcEntity]
||  (nodes[link.SrcEntity] = {name:
link.SrcEntity});
    link.target = nodes[link.DstEntity]
||  (nodes[link.DstEntity] = {name:
link.DstEntity});

    link.value = +link.value;

    });

}

else if (select1.value === "1" &&
select2.value === "5")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcEntity]
||  (nodes[link.SrcEntity] = {name:
link.SrcEntity});
    link.target = nodes[link.DstPort] ||
(nodes[link.DstPort] = {name:
link.DstPort});


    link.value = +link.value;

    });

}

else if (select1.value === "1" &&
select2.value ==="6")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcEntity]
||  (nodes[link.SrcEntity] = {name:
link.SrcEntity});
    link.target = nodes[link.DstCountry]
||  (nodes[link.DstCountry] = {name:
link.DstCountry});
    link.value = +link.value;

    });

}
else if (select1.value === "2" &&
select2.value === "4")  {

    // Compute the distinct nodes from
the links.
```

```
    links.forEach(function(link) {

    link.source = nodes[link.SrcPort] ||
(nodes[link.SrcPort] = {name:
link.SrcPort});
    link.target = nodes[link.DstEntity]
||  (nodes[link.DstEntity] = {name:
link.DstEntity});

    link.value = +link.value;

    });

}
else if (select1.value === "2" &&
select2.value === "5")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcPort] ||
(nodes[link.SrcPort] = {name:
link.SrcPort});
    link.target = nodes[link.DstPort] ||
(nodes[link.DstPort] = {name:
link.DstPort});

    link.value = +link.value;

    });

}

else if (select1.value === "2" &&
select2.value === "6")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcPort] ||
(nodes[link.SrcPort] = {name:
link.SrcPort});
    link.target = nodes[link.DstCountry]
||  (nodes[link.DstCountry] = {name:
link.DstCountry});

    link.value = +link.value;

    });

}

else if (select1.value === "3" &&
select2.value === "4")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcCountry]
||  (nodes[link.SrcCountry] = {name:
link.SrcCountry});
    link.target = nodes[link.DstEntity]
||  (nodes[link.DstEntity] = {name:
link.DstEntity});

    link.value = +link.value;

    });

}
else if (select1.value === "3" &&
select2.value === "5")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcCountry]
||  (nodes[link.SrcCountry] = {name:
link.SrcCountry});
    link.target = nodes[link.DstPort] ||
(nodes[link.DstPort] = {name:
link.DstPort});

    link.value = +link.value;

    });

}

else if (select1.value === "3" &&
select2.value === "6")  {

    // Compute the distinct nodes from
the links.
    links.forEach(function(link) {

    link.source = nodes[link.SrcCountry]
||  (nodes[link.SrcCountry] = {name:
link.SrcCountry});
    link.target = nodes[link.DstCountry]
||  (nodes[link.DstCountry] = {name:
link.DstCountry});

    link.value = +link.value;

    });

}

var width = 960,
    height = 500;
```

```
var force = d3.layout.force()
    .nodes(d3.values(nodes))
    .links(links)
    .size([width, height])
    .linkDistance(60)
    .charge(-300)
    .on("tick", tick)
    .start();

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

// build the arrow.
svg.append("svg:defs").selectAll("marker"
)
    .data(["end"])      // Different
link/path types can be defined here
  .enter().append("svg:marker")    //
This section adds in the arrows
    .attr("id", String)
    .attr("viewBox", "0 -5 10 10")
    .attr("refX", 15)
    .attr("refY", -1.5)
    .attr("markerWidth", 6)
    .attr("markerHeight", 6)
    .attr("orient", "auto")
  .append("svg:path")
    .attr("d", "M0,-5L10,0L0,5");

// add the links and the arrows
var path =
svg.append("svg:g").selectAll("path")
    .data(force.links())
  .enter().append("svg:path")
//    .attr("class", function(d) { return
"link " + d.type; })
    .attr("class", "link")
    .attr("marker-end", "url(#end)");

// define the nodes
var node = svg.selectAll(".node")
    .data(force.nodes())
  .enter().append("g")
    .attr("class", "node")
    .on("click", click)
    .on("dblclick", dblclick)
    .call(force.drag);

// add the nodes
node.append("circle")
    .attr("r", 5);

// add the text
node.append("text")
    .attr("x", 12)
    .attr("dy", ".35em")
    .text(function(d) { return d.name;
});

// add the curvy allTextLines
function tick() {
    path.attr("d", function(d) {
        var dx = d.target.x - d.source.x,
            dy = d.target.y - d.source.y,
            dr = Math.sqrt(dx * dx + dy *
dy);
        return "M" +
            d.source.x + "," +
            d.source.y + "A" +
            dr + "," + dr + " 0 0,1 " +
            d.target.x + "," +
            d.target.y;
    });

    node
        .attr("transform", function(d) {
        return "translate(" + d.x + "," +
d.y + ")"; });
}

// action to take on mouse click
function click() {

d3.select(this).select("text").transition
()
        .duration(750)
        .attr("x", 22)
        .style("fill", "steelblue")
        .style("stroke", "red")
        .style("stroke-width", ".5px")
        .style("font", "20px sans-
serif");

d3.select(this).select("circle").transiti
on()
        .duration(750)
        .attr("r", 16)
        .style("fill", "red");
}

// action to take on mouse double click
function dblclick() {

d3.select(this).select("circle").transiti
on()
        .duration(750)
        .attr("r", 6)
        .style("fill", "#ccc");

d3.select(this).select("text").transition
()
        .duration(750)
        .attr("x", 12)
        .style("stroke", "none")
```

```
        .style("fill", "black")
        .style("stroke", "none")
        .style("font", "10px sans-
serif");
}
```

```
});
}

</script>
</body>
</html>
```