Cyber Security Exercise Modeling & Tracking

Building RESTful Service and MVC Web Application for Visualization and Tracking of Cyber Security Exercise Execution using Modern Web Techniques and Standards

Joonas Greis

Bachelor’s thesis
May 2016
Software Engineering
Bachelor's Degree in Software Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s) Greis, Joonas	Type of publication Bachelor’s thesis	Date May 2016
		Language of publication: english
	Number of pages 33	Permission for web publication: x

Title of publication
Cyber Security Exercise Modeling & Tracking
Building RESTful Service and MVC Web Application for Visualization and Tracking of Cyber Security Exercise Execution using Modern Web Techniques and Standards

Degree programme
Bachelor's Degree in Software Engineering

Supervisor(s)
Luostarinen, Hannu
Hämäläinen, Raija

Assigned by
JYVSECTEC
Silokunnas, Marko
Niemelä, Antti

Abstract

JYVSECTEC has a cyber range where cyber defense exercises are held. Presently there was no graphical interface for participating teams to visualize CDX networks for tagging threats to them in the exercises. This visualization system was designed to meet that requirement and be used as a part of the reporting process.

The objective was to develop for visualizing the range a web based system to which the teams could enter information and metadata on threats they think they notice during the exercise. Each team would have their own perspective and view of the cyber range and they should be able to modify the network to look like they think it would look. Changes on one team's view should propagate to other clients's views in real time.

The System was implemented using modern web techniques and the newest standards. RESTful API was written with Golang using Go-Json-Rest package and the visualization framework was built using Vis.js graph visualization library on top of MVC architecture backbone.js library. PostgreSQL was selected as database.

Keywords/tags (subjects)
Cyber Security, Cyber Range, Cyber Exercise, Network, Modeling, Tracking, Visualization, Simulation, REST, RESTful, Web Service, Web Application, Model-View-Presenter, MVP

Miscellaneous

Kuvailulehti

Tekijä(t) Greis, Joonas	Julkaisun laji Opinnäytetyö, AMK	Päivämäärä May 2016
		Julkaisun kieli: Englanti
	Number of pages 33	Verkkojulkaisulupa myönnetty: x

Työn nimi
Cyber Security Exercise Modeling & Tracking
Building RESTful Service and MVC Web Application for Visualization and Tracking of Cyber Security Exercise Execution using Modern Web Techniques and Standards

Tutkinto-ohjelma
Ohjelmistotekniikka, AMK

Työn ohjaaja(t)
Luostarinen, Hannu
Hämäläinen, Raija

Toimeksiantaja(t)
JYVSECTEC
Silokunnas, Marko
Niemelä, Antti

Tiivistelmä

JYVSECTEC:illä on kybertoimintaympäristö jossa järjestetään kyberturvallisuusharjoituksia. Mitään valmista tai käytössä olevaa graafista käyttöliittymää, johon osaaottavat joukkueet voisivat luoda kuvaa harjoituksen tietoverkosta ja merkitä tapahtumia ja hyökkäyksiä, jotka he luulevat havaitsevansa. Visualisointijärjestelmä kehitettiin täyttämään tämä tarkoitus.

Tarkoitus oli kehittää www-pohjainen järjestelmä kybertoimintaympäristön tietoverkon visualisointiin, minne joukkueet voisivat täydentää tietoja ja uhkia mitä he luulevat huomaavansa. Jokaisella joukkueella olisi oma näkymä kybertoimintaympäristöstä, jota he muokkaisivat mielensä mukaan sellaiseksi, kuin he verkon näkevät. Muutokset jonkin joukkueen näkymään pitäisi päivittyä muihin kytkeytyneisiin näkymiin reaaliajassa.

Järjestelmä toteutettiin käyttäen uusimpia www-tekniikoita ja standardeja. RESTful sovellusrajapinta kirjoitettiin Golangilla käyttäen Go-Json-Rest pakettia ja visualisointi rakenne toteutettiin integroimalla Vis.js kaavioiden piirtokirjasto MVC arkkitehtuurin toteuttavan backbone.js kirjaston päälle. Tietokannaksi valikoitui PostgreSQL.

Avainsanat (asiasanat)
Kyberturvallisuus, kyberharjoitusalue, kyberharjoitus, tietoverkko, mallinnus, seuranta, visualisointi, simulointi, REST, RESTful, Web-palvelu, Web-sovellus, MVP

Muut tiedot

Content

Figures

Codes

Terminology

AAR	After Action Report
CDX	Cyber Defense Exercise
CIDR	Classless Inter-Domain Routing
Cyber Range	Cyber Exercise Environment
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
FPC	Final Planning Conference
Global Event	Disaster, etc.
Hotwash	Debrief conducted immediately after an exercise
HTTP	HyperText Transfer Protocol
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
Incident	Violation or threat of violation of computer security policies
Inject	Specific activity executed as part of a MSEL
IoT	Internet of Things
IPC	Initial Planning Conference
ISP	Internet Service Provider
MitM	Man in the Middle
MPC	Main Planning Conference
NTP	Network Time Protocol
POP3	Post Office Protocol 3
SMTP	Simple Mail Transfer Protocol
UI	User Interface
XMPP	Extensible Messaging and Presence Protocol

1 Visualization and Reporting System for Cyber Exercises

In this thesis a visualization and reporting system was developed. The project was assigned by JYVSECTEC.

JYVSECTEC has a cyber range where cyber defense exercises are held. There is presently no graphical interface for participating teams to visualize CDX networks for tagging threats to them in the exercises. This visualization system is designed to meet that demand and be used as a part of the reporting process.

2 Background

Cyber security is a fast growing industry at the moment. According to MarketsandMarkets' report from 2014 the global cyber security market revenue will double in the next five years. With the IoT becoming more common the cyber security will play a more significant role in the society. (Baxter, A. 2014)

Finland wants to invest in the cyber security industry now and Central Finland is taking part in this with JYVSECTEC's cyber range for research, development, training and exercise. (Limnéll, J. 2016)

Cyber security industry needs more talent and creative development as it grows and this thesis will try to provide both of those.

2.1 Cyber Security

There is no universally accepted definition of Cyber Security, even though term may seem self-explanatory. Cyber Security can be seen as protection of information systems from all kinds of malicious actions against them, like theft or damage to the hardware, the software, or to the information, regardless of the type of action being intentional or accidental. (Refsdal,Solhaug & Stølen 2015, 29-30.)

There are backdoors, DDOS, MitM, spoofing, tampering, phishing, priviledge escalation and more ways to do harm as well as all the natural disasters from broken hard drive to earthquakes. (Mt.)

2.2 Locked Shields

Locked Shields is the largest global cyber defence exercise, organised annually since 2010 by the NATO Cooperative Cyber Defence Centre of Excellence. It is held in the Estonian Cyber Range with new attack vectors and technologies being introduced every year. Locked Shields uses realistic technologies, networks and attack methods. In addition to technical and forensic challenges, Locked Shields also includes media and legal injects providing insight how complex a modern cyber defence crisis can be, and what is required from nations in order to be able to cope with these threats. (After Action Report 2013)

After Action Reports from Locked Shield events give great amount of information on the general state of nations cyber defense. Each year the Blue Teams have become more skilled and better prepared competing in the exercise. More complex and scenario specific technical environments are needed to reflect more closely to real world challenges. (Mt.)

It gives a great picture of the whole scope of exercise and feedback from areas where improvement is needed and what are the key points in defeating cyber attacks. Good monitoring skill was the key capability required to defeat the Red Team as well as good teamwork and communicating skills. Good understanding of the cyber range network helps to convey the big picture. With good communication channel it helps also coordinating defensive operations and measures. (Mt.)

This thesis tries to answer some of those challenges creating a tool for visualizing the cyber range network and using that visual model to tag and share information between the essential personnel.

2.3 JYVSECTEC

Jyväskylä Security Technology project was started in JAMK University of Applied Sciences in September 2011 to meet the growing needs of cyber security research and development. The motivation was to build the most advanced research and development and training centers to Central Finland and bring national and international actors together and form a co-operation network with them. (About us n.d.)

The main product of JYVSECTEC is Realistic Global Cyber Environment (RGCE), a cyber range for research, development, training and exercise. Jyväskylä Security Technology also creates, maintains and cares for up-to-date and impartial cyber security information, as it polishes the image of Finnish cyber security expertise. (Cyber Environment n.d.)

2.4 JYVSECTEC – RGCE

Realistic Global Cyber Environment (RGCE) is an isolated and controlled environment that mimics the Internet. It functions just like the Internet and simulates real network traffic from emails to video-streaming. Cyber environment is used to research and development as well as training the target audience in exercises. (Mt.)

The environment can model multiple ISPs of different sizes, and with public IP addresses and real geographic locations. The modeled ISPs implement Internet core services like DNS, NTP, and of course the web-services. (Mt.)

The network traffic is mostly produced with bots that can be controlled via web based UI. Bot themselves work like a botnet and each bot has its own role and functionality. Fox example, there is a web crawler bot that simulates internet browsing and HTTP requests, a mailer bot, that sends and receives emails using IMAP or POP3 and SMTP, and a communication bot, that simulates chatting over different protocols, like Jabber or Facebook (XMPP). Bots are launched against target services where traffic is generated. There are several ready targets included in the RGCE. All the main web services are covered from email- and time services to controlled update repositories and social media. (Mt.)

The JYVSECTEC cyber range was implemented as a so called hybrid model, where some parts of the network are virtualized and some parts are built with real physical devices. Using hybrid model gives more flexibility over building the environment compared to full physical model and having some physical devices also makes the exercise feel more genuine that it would be if only virtualized. With realistic implementation of customer's organization environment it is possible to test and evaluate real threats, vulnerabilities and network attacks when faced. (Mt.)

3 Cyber Security Exercise

Cyber security exercise is an event where attacks and vulnerability threats against the trainable audience are put to test and evaluated. It consists months of planning and implementing the environment, as it accumulates to the few days' long exercise itself. (Kick 2014)

3.1 Teams

A full size cyber security exercise consists of five teams, White Team (WT), Red Team (RT), Blue Team (BT), Green Team (GT), and an Exercise Control Group (ECG). However, team usage differs with organizers. JYVSECTEC and Locked Shields do not use ECG at all. The role of the ECG has been moved to WT and GT is more like the team responsible holding the range infrastructure together. They fix network problems and make sure that all the network devices run smoothly. While ensuring that the cyber range functions correctly GT may also be responsible for background noise and normal network traffic. This is usually automated. Locked Shields also use a Yellow Team, that maintains the situational awareness and is overlapping with the White Team role. (After Action Report 2013; Kick 2014)

3.1.1 Exercise Control Group

Exercise Control Group (ECG) works as a controller of the exercise as they are the only ones that have the event manuscript, Master Scenario Event List (MSEL) that contains all planned attacks and injects to be executed. ECG drives the execution by giving tasks to Red Team and injects global events that are scheduled to happen. (Mt.)

3.1.2 White Team

White Team collects information as the exercise progresses and provides feedback to the control group and Red Team. They work as observers and solve possible conflicts that may arise between the red and blue teams as well as answers questions that may arise during the exercise. White Team also evaluates the team progression by assigning scores to teams. (Mt.)

3.1.3 Red Team

Red Team is the perpetrator that executes planned injects and attacks against the Blue Teams. Its objective is to improve the target company cyber assurance by demonstrating the impacts of cyber-attacks on defenders enterprise system. (Mt.)

3.1.4 Blue Team

Blue Team is the most important actor in cyber exercise as it consists the training audience. Their role is to be the defenders in the exercise and keeping the enterprise information systems secure. There can be multiple Blue Teams that form together the company network. (Mt.)

3.1.5 Green Team

There are two common definitions for Green Team. In some exercises the Green Team represents normal network traffic and background noise, and some exercises use it as a technical team, who are responsible for preparing and maintaining the cyber range technical infrastructure. Green Teams that represent common user traffic are usually automated. JYVSECTEC uses Green Team for ensuring the cyber range infrastructure is operational. (Kick 2014; Silokunnas 2016)

3.2 Planning

The structure and planning of a cyber exercise are similar across different organizations. Every exercise starts with planning across the participants. Exercise scenario is formed where the strategic and operating environment are described in sufficient scope and detail. The expected reactions of training audience are also included in the exercise scenario. In practice, it is the storyline or plot for the entire exercise, utilized by all planners and participants. (Mt.)

There are three types of exercises. Table top exercise is the least complex and fastest to execute. It is a paper-driven exercise where injects and attacks are scripted via paper. Hybrid exercise uses some realism in the attacks while leaving some injects to be delivered with pen and paper. Full Live is delivered fully with real injects in real-like environment. These are the most complex and largest exercises to be held and it takes a year to plan a full live exercise. (Kick 2014)

3.3 Execution

After months of planning and environment building the exercise is ready to be started. The exercise is usually divided in phases or sessions. Each session has a different portion of scenario flow, so training audience can focus on one ensemble at the time. (Cichonski, Millar, Grance & Scarfone 2012)

ECG drives the execution according the MSEL and tasks RT who is the perpetrator of the exercise. WT observes and solves possible conflicts that may arise during the exercise. (See Figure 1)

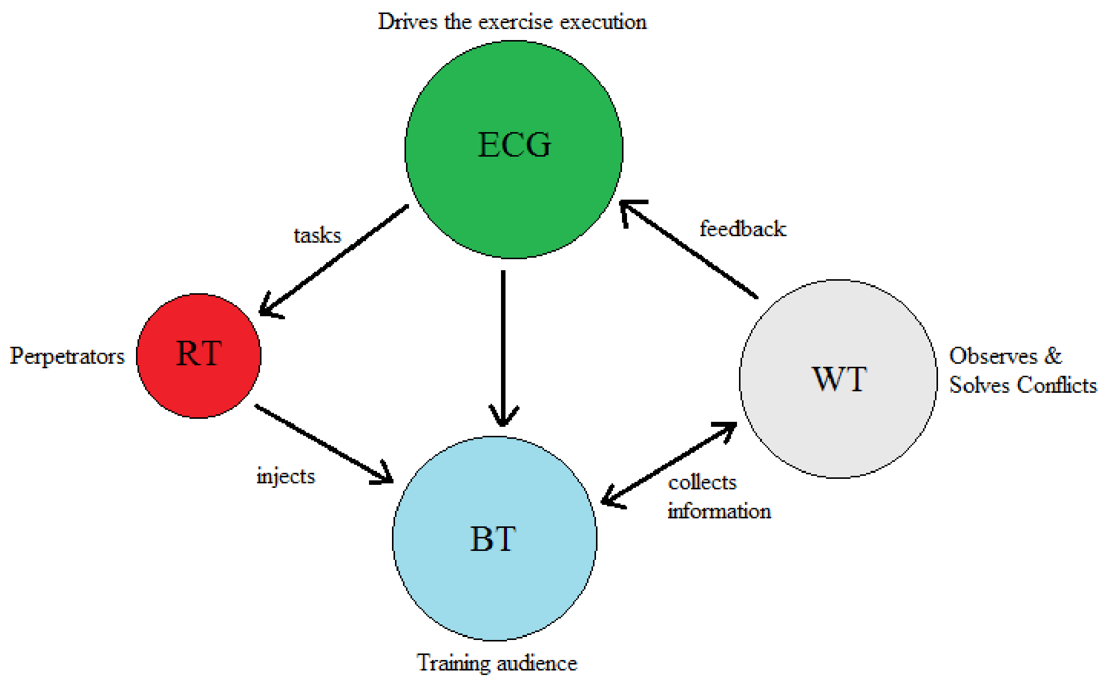


Figure 1. Exercise Flow

Provided the planning was thorough, the execution is merely about following the exercise plan and monitoring the training audience responses. (Kick 2014)

3.4 Post Exercise

Immediately after the exercise a hotwash is held. Hotwash is a debrief conducted with staff and participants. More profound retrospective analysis is done later for the After Action Report (AAR). These can be used to improve the training audience company security policies and ability to respond to different cyber threats. Reports are also used to improve the cyber exercise itself. (Kick 2014)

4 Project Objectives and Requisites

Before this thesis, the exercise tracking and reporting were done with text based chat. Thus there was a great need for cyber range visualizing system for tracking actions and incidents in the network and reporting them.

Team View reporting tool would be a great aid in collecting the exercise data and synchronizing different teams' reports. It could also be used as a part of AAR in analyzing the exercise progression and Blue Teams responsiveness to Red Teams actions.

4.1 Team View

There was a need for a system, where each team could visualize their own perspective of the network, map metadata and report incidents. Each team should have its own view of the cyber range network, where they could easily add elements and connections of choice.

Blue Teams would have their own company networks mapped with few elements from outside the company network as well. And because companies are usually compartmentalized, each Blue Team would have its own separate view of the related part of the network with possibly overlapping regions with others.

Red Team view would be a bit larger subset of the network as it usually contains all the blue team views as well as the red team's own devices and interfaces.

The largest network perspective would be seen by White Team and ECG as they both should be able to monitor the exercise progression.

4.2 Customer Requirements

Now that the project subject was developing a visualization system for cyber security exercise tracking and reporting, it was time to define the customer requirements.

System should be cross-platform and use modern web techniques and standards.

There should be different views for Red Team, White Team and Blue Teams so the changes that blue team will make will show up on Red Teams view, but the changes on Red Team view will nott appear on the Blue Team View. Changes should also be propagated to the other views almost in real time.

All changes should have a timestamp for network history implementation and network should be able to be viewed in any state and time wanted.

BasicAuth will be enough for user validation. There is no need for more secure and more complex authentication method, because the system will be run only on isolated cyber range.

The system should also produce a base network map from network data given in JSON format for teams to use as a template in visualizing their own perspective.

There should be three different topology view modes implemented, physical topology, logical topology and a hybrid of these two topologies, which seems to be the most popular choice of these three. Different topologies should be able to be toggled as the metadata bound to elements and views stays the same.

There is also a need for icons and highlighted elements for different meanings, like marking some node as "vulnerable" or "infected".

And as it will be a reporting tool, reporting must be also possible and the final report should be able to export from the system to be used in exercise after review.

5 Related Work

There are few similar cyber security related systems, but this is the only web-based cyber range network visualization and reporting tool there is.

Most of the systems found on public search engines are complex and comprehensive state-funded systems. They require to be installed in very specific cyber ranges.

The lack of these kinds of systems and applications is strange when reviewing the Locked Shield After Action Reports where many of them state the fact that there is lot to improve in cyber range network awareness and communication and delegation channels. (After Action Report 2012; After Action Report 2013)

6 Design

6.1 Issues and Questions

There were many things to be considered during the preplanning phase. For being a first web-based cyber exercise visualizing and reporting tool few issues and thoughts first had to be solved. For example, how to present different topology views and bind the metadata to elements in the views? Or how to tag Cloud DDOS, BGP MitM, ARP Spoofing, and other attacks that utilize outside the cyber range view, as well as to display information on clouds, social media, factories, natural disasters and other non-generic elements and data?

Requirements define the techniques needed to use. The graphic visualization would be the bottleneck of the system so information about lightweight visualization libraries and techniques was needed.

6.1.1 WebGL, Canvas or SVG?

The initial idea was to use WebGL for graphics, so it would support ridiculously large networks. However, after learning that usual exercise network has only under 100 nodes, doing WebGL graphics started to feel like overkill.

SVG, on the other hand, is bound to DOM, so even with the network of just few nodes, the web application comes a bit slow and jerky. SVG has its advantages though in CSS styles and -animation, that offer a powerful way to modify the SVG elements.

HTML5 Canvas was selected. It has enough power for medium-sized networks and custom elements can be easily drawn to container. It is also much simpler to use than WebGL graphics.

6.1.2 Architectural Patterns

What would be the best approach in developing this kind of visualizing and reporting web application? How will the data be moved across the application network? What kind of application architecture would work best?

The basic picture started to form fast. A database would be needed to hold all the network data. An API that works as a controller between the application back-end and the database was needed as well. The application needed its own structure also. (See Figure 2)

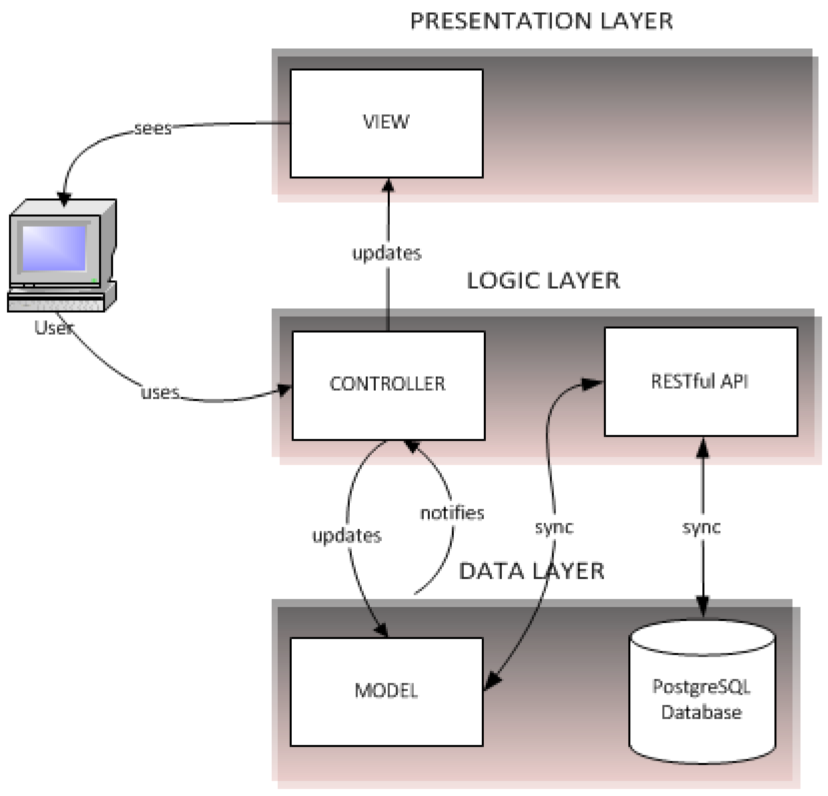


Figure 2. Architectural Model

The database should be document-oriented database or some other NoSQL database because of the data structure.

And since there must be multiple simultaneous clients connected with different views RESTful service was chosen to be implemented between the application layer and the database.

Because a web-based visualization system with different views on the same data was to be built, the only viable options for application architecture were MVVM, MVC, MVP and similar patterns. It really was not a major decision because the main point was to have separate views from same data with different logic, so all the mentioned patterns would do just fine.

6.2 Visualization Layer Back-end

The choice of back-end became easy after the criteria were defined. A light and modular JavaScript framework was needed that would support MVC or MVP application design paradigm with a RESTful JSON interface. There was no need for a complete stack, like Node.js or Done.js. Also frameworks like React and Ample SDK were way too heavy for to be suitable for this kind of use. Two frameworks stood out, backbone.js and Ambersand.js, both lightweight and modular frameworks. Backbone.js was selected due the integrated RESTful JSON syncing.

Backbone.js is a MVP (or MVC) style architecture Java-Script library with integrated RESTful JSON interface. It is created by Jeremy Ashkenas, who is also known for CoffeeScript and Underscore.js fox example. And Underscore.js is one (and only) dependency of Backbone.js. (Ashkenas 2016)

6.3 Visualization Layer Front-end

As well as all the other libraries used the graphic library should also be open source. The first choice was Sigma.js that has superior performance in drawing network graphs compared to other libraries because it uses WebGL for rendering. It was also the most modular and lightweight library from the choices of mine.

The reason Sigma.js was dropped was that it would be so much easier to customize Canvas graphics than WebGL graphics, now that it was known that there was no need for thousands of nodes. Sigma.js also supports HTML5 Canvas but after some testing it was realized that the Canvas renderer was not as fast as Canvas renderer in Vis.js, the second option for network graph visualizing library. Vis.js was also slightly more advanced and had more predefined functionalities like clustering and grouping. (Almende 2016)

There are also several more powerful visualization libraries, like Vega and D3 for example, however they are too powerful for this kind of a project. (Bostock 2015)

Vis.js is a JavaScript visualization library designed to be easy to use, to handle large amounts of data, and enable data manipulation. In consists of five different components but only DataSet and Network components are used to create the network visualization. (Almende 2016)

6.4 Networking

Again, the most significant factor is how much static and dynamic data has to be handled, and how often information have to be exchanged between the client and the server. Because only an update of the views is needed when someone pushes a change to data, there is no need for continuous TCP connection like in WebSockets. The fastest way would have been to implement custom Server Side Events with XMLHttpRequests, where SSE would have updated the client views when server was changed and HTTP XHR would have been used to send data to the server. However, Backbone.js uses jQuery.ajax to make a RESTful JSON request by default and returns a jQuery XMLHttpRequest (jqXHR) object, which is a superset of the browser's native XMLHttpRequest object. Ajax requests pursue their case, so there was no need to override the functionality. If the application networking needs will grow, the communication method can easily be changed to more efficient one. Like if there was a need for real live syncing with the views, WebSocket or SSE would be only viable option. (jQuery API Documentation 2016; Sheiko 2012)

6.4.1 RESTful Service

Golang seemed like a good programming language of choice for building RESTful API. It is easy to write concurrently running code by using goroutines. Also there is a golang package Go-Json-Rest which is a thin layer on top of net/http package that helps building RESTful JSON APIs easily. (Effective Go 2016; Imbert 2016)

6.5 Database

The initial candidate was MongoDB, an open-source document-oriented database, however,- JYVSECTEC uses PostgreSQL with their systems so they wanted to use it with this as well. PostgreSQL is, instead, an object-relational database and not an NoSQL database at all. Both of them support JSON documents with key - value pairs. The reason to choose MongoDB over PostgreSQL is scalability. MongoDB is horizontally scalable and PostgreSQL is vertically scalable, so MongoDB could easily be scaled out if needed, -however, data integrity might be weaken. (Mohammed 2015)

Every data-node must contain the time of creation field and a new record of the node must be created when updating any of the nodes key - value pairs, so the state of the network can be queried and displayed according the elapsed time. (See Figure 3)

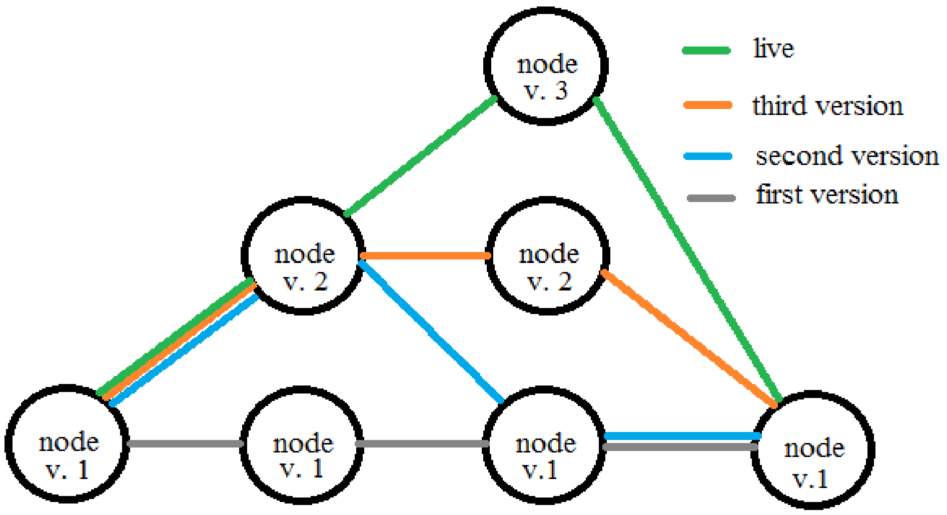


Figure 3. Node History Storing Model

6.6 Data Import

Data import should be possible in JSON format with RESTful API straight to the database. Imported data could then be used to create the base network for team's view.

6.7 Standards

Because newest standards should be followed, the application should use ECMASCRIPT 2015 (ES6) in JavaScript logic and HTML5 in web elements. Also network Icons should be simple and recognizable, so Cisco network topology icons should be used. They are globally recognized and generally accepted as a standard for network icon topologies. (Network Topology Icons – Doing Business With Cisco 2016)

This thesis is also written according academic standard and Raija Hämäläinen supervises the process.

7 Implementation

7.1 Nodes

Nodes hold the element information and metadata bound to it. They are extended backbone.Models with element data and visualization settings and hold at least the key-value pairs in vis.js Network component (See Code 1)

```
var Node = Backbone.Model.extend({
      defaults: {
            Id: null,
            Name: null,
            IPv4: null,
            Label: null,
            type: 'node',
            image: 'img/cisco/Host.png',
            shape: 'image',
            font: {
                  face: 'Arial',
                  color: '#000'
            },
            shadow: {
                  enabled: true,
                  color: '#000'
            },
            title: null,
            group: null,
            x: null,
            y: null,
      }
      });
```

Code 1. Model

Nodes usually belong to a zone, however it is not mandatory.

7.2 Zones

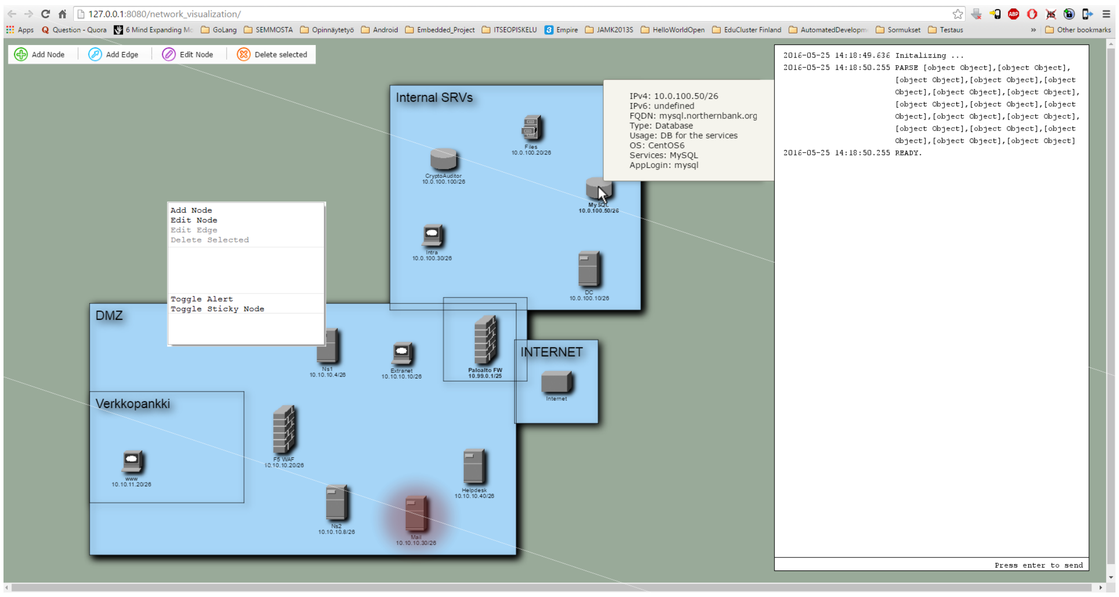
Zones are logical segments of network. They may base on logical or physical topology, but not necessarily. They help the layout and displayi the nodes in the network and visualize the connections between the nodes. Zone can have multiple parent zones and vice versa.

7.3 Visualization Layer

Visualization system is an easy to use web application where users can edit network structure and elements, add metadata to them and toggle threats without having the need to write long descriptive texts to reporting chat.

Chat functionality is nonetheless integrated in the web interface.

A working prototype was built first. (See Figure 4)



```
IPv4: 10.0.100.50/26
IPv6: undefined
FQDN: mysql.northernbank.org
Type: Database
Usage: DB for the services
OS: CentOS6
Services: MySQL
AppLogin: mysql
```

```
2016-05-25 14:18:49.636 Initalizing ...
2016-05-25 14:18:50.255 PARSE [object Object],[object Object],
                        [object Object],[object Object],[object
                        Object],[object Object],[object Object],
                        [object Object],[object Object],[object
                        Object],[object Object],[object Object],
                        [object Object],[object Object],[object
                        Object],[object Object],[object Object]
2016-05-25 14:18:50.255 READY.
```

Figure 4. Prototype Application

7.3.1 Visualization Framework Directory Structure

(See Appendice 1)

.config.cfg file that is found from the root directory is used to define credentials for the RESTful API and some basic configurations about the graph physics and looks.

7.3.2 Session Handling and Local Storage

Because using the newest and most modern techniques, HTML5 Local Storage is used for storing user- and session variables, so when restarting browser session and navigating back to visualization site, the network layout remains the same as it was before exiting.

HTML5 Session Storage is similarly used in storing session variables that are required only during the current session.

7.3.3 Data Relations

Persistent data is stored to PostgreSQL database and is accessed over HTTP via RESTful interface.

Runtime data is stored in backbone.Collection that consists of extended backbone.Models customized for network graph drawing and metadata handling. (See Figure 5)

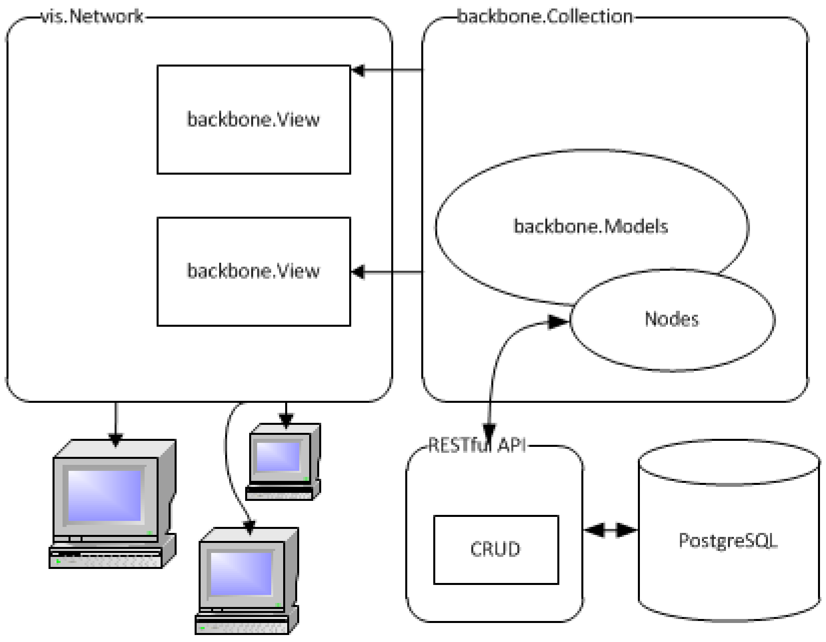


Figure 5. Data Relations

backbone.Views are used define the visible elements for each teams perspective. Data from the view is bound to vis.Network element where vis.View filters are used to display the network state at given time. The timeline moving logic was preferred to be kept on the client side with local data, so moving along the timeline will be fast and smooth.

7.3.4 Rendering View

Rendering function is as simple as possible. It is set to run every time the network data changes or data is fetched from the server. (See Code 2)

```
this.listenTo(network, 'change', this.render);          //event listener
this.listenToOnce(network, 'sync', this.render);        //event listener
```

Code 2. Listening Events

Rendering function updates the vis.js network graph, and the view it is bound to, automatically populates itself. (See Code 3)

```
render: function() {
      network.forEach(generateEdges);
      network_graph.setOptions(options.View);
      network_graph.setData({
      nodes:network.toJSON(),
      edges:generated_edges
      });
      return this;
}
```

Code 3. Render Function

Function generateEdges is custom function that parses and calculates node relations and generates a map of relations between the nodes using the CIDR and Zone defined in each node. Generated edge-set is used to simulate network graph physics. For example the nodes on the same zone are bound together and when dragging one of them, the others will follow in accordance with the physics model.

Zone drawing and zone relation computations are done on beforeDrawing function thus giving more performance to the graph visualizer, when not calculating zone data every time the scene is rendered.

Custom drawing and visualization is done with extended and highly customized Vis.js library.

7.4 Backbone

Backbone.js works as a backbone for the application.

It implements functionalities for adding and modifying the nodes, edges and zones.

Every time when user submits a change, the backbone launches jQuery.ajax request to RESTful API for updating the network data.

7.5 RESTful API

RESTful API was build with Golang and it uses Go-Json-Rest package in implementing the RESTful service. It uses Golang default workspace structure and can be build with golang build tools. (See Appendix 2)

It uses golangs default syslog output method, so the implementation works only on linux for now.

One of the biggest advantages of using Go-Json-Rest package is ability to easily setup and configure CORS middleware around the API endpoints. (See Code 4)

```
api.Use(rest.DefaultDevStack...)
       api.Use(&rest.CorsMiddleware{
               RejectNonCorsRequests: false,
               OriginValidator: func(origin string, request *rest.Request) bool {
                       return origin == host
                },
                AllowedMethods: []string{"GET", "POST", "PUT"},
                AllowedHeaders: []string{"Accept", "Authorization", "Content-Type",
"X-Custom-Header", "Origin"},
                AccessControlAllowCredentials: true,
                AccessControlMaxAge:           3600,
    })
```
Code 4. Using CORS Middleware

Basic Auth was also setup as a pre-routing middleware.

The service implements basic CRUD operations.

Golang package pq was used to connect RESTful API to PostgreSQL database. it is a pure Go PostgreSQL driver for the default database/sql package.

8 Conclusions

There are many different views on cyber exercises how they should be executed. However, they all rely on similar principles on attacking, defending and observing.

There is still research and development to do in finalizing the visualization system and tweaking it up to work with the JYVSECTEC operation environment and package management. Keeping the produced system modular and extendable will help it adapt to JYVSECTEC's cyber operation environment, as well as in keeping up with the changes in other systems. Resorting to standards also makes it more approachable.

It seems like the cyber security as an industry is in a constant change and lots of new approaches are invented on daily basis.

8.1 JavaScript Frameworks & Libraries

There are hundreds of JavaScript visualization frameworks and web application architecture libraries to be used in web development. They also evolve with technology and other techniques. There are open source and proprietary software with all kinds of licenses and there is suitable visualization framework and application structuring library for every project.

8.2 Golang & Packages

Golang is still quite unused language but it was well suited to building RESTful service. Using packages made it easy to implement the middleware and no stumbling was required to configure CORS headers and user authentication.

References

About Us. N.d. Jyväskylä Security Technology. Accessed 24.5.2016. Retrieved from http://jyvsectec.fi/en/about-us/

After Action Report 2012. Locked Shields. Accessed 24.5.2016. Retrieved from https://ccdcoe.org/publications/LockedShields12_AAR.pdf

After Action Report 2013. Locked Shields. Accessed 24.5.2016. Retrieved from https://ccdcoe.org/publications/LockedShields13_AAR.pdf

Antikainen, J. 2014. Model for national cybersecurityresilience and situation awarenessimprovement - An information quality -centric approach leveraging fusion of established practitionerand academic disciplines. Accessed 24.5.2016. Retrieved from www.theseus.fi/handle/10024/86179

API Documentation. N.d. jQuery. Accessed 24.5.2016. Retrieved from http://api.jquery.com/jquery.ajax/

Ashkenas, J. 2016. Backbone.js. Accessed 24.5.2016. Retrieved from http://backbonejs.org/

Baxter A. 2014. Cybersecurity: The Industry That Keeps on Growing. Accessed 24.5.2016. Retrieved from http://www.educause.edu/blogs/vvogel/cybersecurity-industry-keeps-growing

Cichonski, Millar, Grance & Scarfone 2012. Computer Security Incident Handling Guide - Recommendations of the National Institute of Standards and Technology. Accessed 24.5.2016. Retrieved from http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

Concurrency. N.d. Effective Go. Accessed 24.5.2016. Retrieved from https://golang.org/doc/effective_go.html

Cyber Environment. N.d. Jyväskylä Security Technology. Accessed 24.5.2016. Retrieved from http://jyvsectec.fi/en/cyber-environment/

Cybersecurity - Situation Awareness. N.d. Mitre Corporation (?). Accessed 24.5.2016. Retrieved from https://www.mitre.org/sites/default/files/publications/pr_14-3929-cyber-exercise-playbook.pdf

Data-Driven Documents. N.d. D3. Accessed 24.5.2016. Retrieved from https://d3js.org/

DataSet. N.d. Vis.js. Accessed 24.5.2016. Retrieved from http://visjs.org/docs/dataset/

DataView. N.d. Vis.js. Accessed 24.5.2016. Retrieved from http://visjs.org/docs/dataview/

Imbert A. 2016. Go-Json-Rest. Accessed 24.5.2016. Retrieved from https://github.com/ant0ine/go-json-rest

Limnéll, J. 2016. Kyberturvallisuuden tulevaisuus ja kybersodankäynti. Helsinki Engineers HI Association Kyberturvaa! -seminar 6.4.2016. Retrieved from https://www.youtube.com/watch?v=_bIjOTvAltg

Limnéll, J. 2016. Suomesta kyberturvallisuuden tulevaisuus. Accessed 24.5.2016. Retrieved from http://www.aalto.fi/fi/current/news/2016-05-04-002/

Mohammed J. 2015. Is Postgres NoSQL Better Than MongoDB? Accessed 24.5.2016. Retrieved from http://www.aptuz.com/blog/is-postgres-nosql-database-better-than-mongodb/

Network. N.d. Vis.js. Accessed 24.5.2016. Retrieved from http://visjs.org/docs/network/

Package pq. N.d. GoDoc. Accessed 24.5.2016. Retrieved from https://godoc.org/github.com/lib/pq

Sheiko D. 2012. WebSockets vs Server-Sent Events vs Long-polling. Accessed 24.5.2016. Retrieved from http://dsheiko.com/weblog/websockets-vs-sse-vs-long-polling/

Silokunnas, M. 2016. Opparista. Sähköpostiviesti 25.5.2016.

Appendices

Appendix 1. Visualization Framework Directory Structure

```
network_visualization/
      .config.cfg                   # hidden config file
      public_html/
            css/
                  common.min.css    # application css
                  vis.min.css       # vis.js css
            img/
                  cisco/            # cisco network topology icons
                  ..
                  favicon-XXxXX.png # favicons for tablets etc.
                  ...
                  favicon.ico       # favicon
            js/
                  app/
                        app.js      # application main logic
                        common.js   # common functions
                        controls.js # ui controls, edit node etc.
                        render.js   # rendering function
                        view.js     # view handler for different views
                  backbone.js/
                        backbone-min.js         # backbone js
                        backbone.basicauth.js   # basic auth plugin
                  jquery/
                        jquery-2.2.0.min.js     # jquery js
                  underscore.js/
                        underscore-min.js # backbone dependency
                  vis.js/
                        vis.min.js        # vis js
            index.html                    # index html
```

Appendix 2. RESTful Service Directory Structure

```
bin/
      rest                          # command executable
      config.cfg                    # configuration file
      .password                     # password example
pkg/
      linux_amd64/
            jyvsectec.fi/csemtt/rest-go/rest/
                  crud.a            # package object
src/
      jyvsectec.fi/csemtt/rest-go/rest/
      .git/                         # Git repository metadata
      crud/
            crud.go                 # package source
            crud_test.go            # test source
      rest/
            main.go                 # command source
            main_test.go            # test source
```