OpenVPN SDN Management System

# Software Design Document

Team 6

Alan Castle

Keandre Graham

Brianna Howard

Hayes Roach

Chris Strahl

Date: (03/29/2020)

Department: Software Engineering

Course: 4724

Professor: Dr. Parizi

**TABLE OF CONTENTS**

1. INTRODUCTION

1.1 Purpose

1.2 Scope

1.3 Overview

2. SYSTEM OVERVIEW

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

3.2 Technical Design

3.3 Design Rationale

4. DETAILED DESIGN

 4.1 Vue.JS

 4.2 Flask Python

 4.3 MariaDB

 4.4 OpenVPN Server

5. DATABASE (DATA) DESIGN

6. HUMAN INTERFACE DESIGN

7. REQUIREMENTS MATRIX

### INTRODUCTION

## Purpose

This Software Design Specification (SDS) discusses the high-level design of the OpenVPN SDN Management System being created for Cybriant as a senior project by a team of KSU students.

## Scope

The project is meant to allow for Cybriant to quickly and easily set up and configure a network of OpenVPN clients. This solves a need for their business to be able to quickly connect to customers over VPN in a secure manner. This web-based tool will allow Cybriant to identify server IP addresses belonging to particular servers or sets of servers and to configure the connections that may be established between those servers.

## Overview

The first to sections of this document provide an overview of the software and its goals. The 3rd section, an overview of all of the components and how they interact, and all subsequent sections contain discussion on the details of each component.

### SYSTEM OVERVIEW

A helpful way to view the system is to compare the OpenVPN server to a combination router and proxy. Requests for connecting to a specific machine first enter the OpenVPN server, by default an OpenVPN server acts as a router, routing the request to the targeted machine. In our project the OpenVPN server will also check a set of rules to determine if the connection to the target machine is valid and should be allowed, similar to what a proxy does with web traffic. In other words, the OpenVPN server will act as a proxy for outgoing and incoming traffic, determining if the connection that is attempting to be established abides by Cybriant’s configured security rules. If these rules pass, then the connection is routed to the target client.

Much of this functionality is available immediately through the OpenVPN server, however creating and managing these security rules is cumbersome and error prone. To combat this configuration may be done through a web-based GUI provided by this project. This also allows the user to create these configurations at a larger logic-based level rather than having to know the specifics of how to configure each connection for each client. Instead the user will be able to define a server or set of servers (Server Groups) which can be thought of as analogous to an OpenVPN client. By default, any attempted connection with this server group through the server will be denied. The user can then define which server groups can connect to which, along with additional security rules such as which server group(s) is allowed to establish the connection.

### SYSTEM ARCHITECTURE

## Architectural Design



As mentioned previously, much of the functionality needed for the project is already provided by OpenVPN and so the details of accomplishing the proxying and routing are all handled by the OpenVPN server. However, the logical organization of these configurations is being handled by the Flask Python server. How the user interacts with the system is handled by a Vue JavaScript browser-based GUI. Finally, MariaDB is used to store data as needed by the webserver for configuring the OpenVPN network. Additionally, the web server handles authenticating the Cybriant employee using the OAuth2.0 security specification to prevent unauthorized individuals from misconfiguring their network.

## Technical design

Vue.JS is a widespread easy to set up JavaScript framework for quickly building fast and efficient web GUI. Vue.JS is well suited for making RESTful API calls to a backend web server. Flask is also a widespread Python framework for developing RESTful web services. MariaDB was chosen by Cybriant as their preferred SQL flavor. Finally, OpenVPN is a very robust, secure, and well documented VPN service. This detailed documentation makes OpenVPN highly extensible. This paired with Cybriant’s experience with it makes OpenVPN the ideal choice to build the project upon.

## Design Rationale

 Web based clients for controlling systems now provide many more benefits than drawbacks when compared to traditional client installation. A major benefit of having a web-based service makes the system much more portable, being accessible from any machine, even mobile devices. As browser’s efficiency, and therefore speed, have improved the only major benefit a traditional install has is in security. However, through careful planning a web-based client may be very well secured and hardened against attack. For this project, the OAuth2.0 security specification was chosen, not only because of the development team’s familiarity with the specification but also because it is increasingly becoming the industry standard specification.

### Detailed Design

* 1. **Vue.JS Front-end**

 The Vue.JS front-end primarily takes in user input through forms which it then sends in JSON format to the backend to create an entity based on the input. The front-end also receives a list of entities in JSON format which it displays in various tables.

* 1. **Flask Python Back-end**

The Flask Python back-end receives input from the front-end as JSON. It then parses the JSON into a Python Object. The Python Object can then be validated to be good input and saved into the database. Next if applicable Python will generate an OpenVPN rule / configuration based on this input. Flask Python must also compile all active entities in the DB and return them as JSON for the front-end to display

* 1. **MariaDB**

MariaDB must simply serve as any other SQL database, as a set of queryable tables.

* 1. **OpenVPN Server**

As previously mentioned, the OpenVPN server is handling most of the resulting functionality of the system and is an open source component, not one we have built, as such the exact details of how it accomplishes these goals is not clear. Broadly, the OpenVPN server maintains a table called the IPTable in which entries can be created for specific connection rules. For instance, IP A can only connect to IP B on a specific port and can only transfer UDP packets. The system will update this IPTable with configurations as desired by the user.

### DATA (DATABASE) DESIGN

Data can be split into two logical categories, tables for system infrastructure and security, and tables for the actual domain of configuring OpenVPN. The OAuth2.0 specification requires storing data about the user, such as their login info, their role / permissions, etc. as well as records for each of the users active logged in sessions.

For the actual project domain, the main record on which everything is built is the Server Group table. A server group may belong to an organization, which simply serves as categorization by the user. A server group may also have none to many connections either outgoing or incoming. This connection table is where the bulk of the network configuration is stored as each connection specifies an allowed route to the OpenVPN server.

### HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

This project will serve Cybriant in creating Software Defined Networks (SDNs) between infrastructures. The application will specifically allow the client to create uni/bi-directional connectivity to the different infrastructures. All the functionalities of the web application are maintained by the administrators. More precisely, the administrator is able to create, delete, and manage the servers of their clients.

This application has four major screens: the main screen, the user management screen, the network management screen, and the server connection screen. The server connection screen is an interactive visual aid that shows the connection between Cybriants severs and their client’s servers. The main screen allows Cybriant to access all other screens on the application. The network management screen allows Cybriant to create, delete, and edit server groups.

6.2 Screen Images



Network Management Screen



The User Management Screen

### REQUIREMENTS MATRIX

|  |  |  |
| --- | --- | --- |
| **Req ID** | **Requirement Text** | **Associated Component(s)** |
| R1. | The system shall allow the definition of a group of servers. | Flask Python |
| R1.1. | Definition and viewing of server groups shall be available through a web-based GUI. | Vue.JS |
| R1.2. | A group of server’s definitions shall include all information required to connect to a server. This includes but may not be limited to:* IP range
* Virtual IP range
* General Connection Rules and Parameters
 | Flask Python |
| R1.3. | The system shall allow server groups to be put into organizations for server groups that are controlled by the same company or real world entity. | MariaDB, Vue.JS |
| R2. | The system shall optionally be able to create an install package for download when a new server group is defined. | Flask Python |
| R3. | The system shall deny connections between the server groups by default. | OpenVPN Server |
| R4. | The system shall allow the definition of connections between server groups. | Flask Python |
| R4.1. | Definition and viewing of connections shall be available through a web-based GUI | Vue.JS |
| R4.2. | Connection shall include all information required to establish a secure connection between the server groups. This includes but may not be limited to:* Allowed directions of connection
* Connection type allowed (TCP, UDP, SSH, etc.)
* Connection / Request limit
* Specific ports to block
 | Flask Python |
| R4.3. | Connection rules shall be mutable and may be updated without adoption effort for either server group defined in the connection. | OpenVPN Server |
| R5. | The system shall provide a graphical representation of server groups and the connections between them. | Vue.JS |
| R6. | The system shall provide an authentication and log in system to prevent unauthorized access to view or edit items in the system. | Flask Python |
| R6.1. | The system shall provide two authenticated roles one for creating, viewing, editing, and deleting items, and another role for viewing only. | Flask Python |
| R6.2. | The system shall provide a list of users and their role in the web GUI. | Vue.JS |
| R7. | The system shall be usable by the appropriate members of the Cybriant team in an interactive format with little to no delays to input and network usage. | Vue.JS |
| R8. | The system shall be scalable in scope and compatible with Azure servers, with regard to server/client connections, bandwidth, storage, and storage. | Flask Python |
| R9. | The System shall transmit all information securely with no changes to the original information. | OpenVPN Server |
| R9.1. | The connections must follow OpenVPN security protocols. | OpenVPN Server |
| R10. | The system shall be available to users 99% of the time.  | Flask Python |
| R10.1. | The system shall return to full operational status within 15 minutes should an interruption occur.  | Flask Python |
| R11. / R11.1. /R11.2. /R11.3. | The system shall store the following data securely.* Server and Client configuration and updates
* User data, including usernames and passwords, authorization levels, and log data
* Additional data dependent on the growth of scope of the system.
 | MariaDB |