Weave VPN: OpenVPN Without Headache

Test Document

Team 6

Members

Christian Strahl

Alan Castle

KeAndre Graham

Brian Howard

Hayes Roach

Date: 04/19/2020)

Department: CCSE

Course: SWE 4724

Professor: Reza Parizi

**1. Testing Strategy**

1.1 Overall strategy

For the testing, we plan on making sure the basic functionality of the system works, such as having a user log in, download packages to setup OpenVPN on their system, and finally connect to it. On Cybriant side, we will test to make sure they can setup organizations to house server groups and specify the details on the connection pathway of the server groups (bi-directional or uni directional)

1.2 Test Selection

Testing of the application will focus on the target requirements outlined by the client, which can be traced directly to use cases. The goals of these tests are to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the clients requirements. This type of testing is based upon black box techniques, that is, verifying the application (and its internal processes) by interacting with the application via the GUI and analyzing the output (results).

There will be two cycles for functional testing. Each cycle will execute all the scripts . The objective of the first cycle is to identify any blocking, critical defects, and most of the high defects. It is expected to use some work-around in order to get to all the scripts. The objective of the second cycle is to identify remaining high and medium defects, remove the work-around from the first cycle, correct gaps in the scripts and obtain performance results. System testing will be performed before we allow the client to test the application. User acceptance testing will consist of one cycle, which allows end users to complete one final review of the system prior to deployment.

1.3 Adequacy Criterion

Testing is mostly to be done at an integration, or system, level. Testing should verify that all API resources are reachable and function as intended.

1.4 Bug Tracking

Bugs and enhancements can be created and tracked through the project’s GitHub Issues page.

**2. Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Ensure proper application navigation, data entry, processing, and retrieval | Execute each use case, use case flow, or function, using valid and invalid data, to verify the following:  The expected results occur when valid data is used.  The appropriate error/ warning messages are displayed when invalid data is used. | All planned tests have been executed.  All identified defects have been addressed |  |
| Validate System Response time for server connections under the following two conditions:  -normal anticipated volume  -anticipated worse case volume | Modify data files (to increase the number of server connections) or modify scripts to increase the number of iterations each server connection occurs  Scripts should be run on one machine (best case to benchmark single user, single connections) and be repeated with multiple clients (virtual or actual) | Single Connection / single user: Successful completion of the test scripts without any failures and within the expected / required time allocation (per transaction)  Multiple connections / multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation. |  |
| Function/ Data Security: Verify that user can access only those functions/ data for which their user type is provided permissions | Identify and list each user type and the functions / data each type has permissions for.  Create tests for each user type and verify permission by creating transactions specific to each user type.  Modify user type and re-run tests for the same users. In each case verify those additional functions / data are correctly available or denied. | For each known user type the appropriate function / data are available |  |
| Validate and verify that the Application functions properly on the prescribed client workstations | Use Integration and System Test scripts.  Execute selected processes to simulate user activities in the application.  Repeat the above process, minimizing the available conventional memory on the client. | The process is successfully completed without failure |  |
| Ensure the Database accesses methods and processes functions properly and without data corruption | Invoke each database access method and process, seeding each with valid and invalid data (or requests for data)  Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved | All database access methods and processes function as designed and without any data corruption |  |

**3. Test Results**

1. The result of the first use case was successful. Navigation worked as intended, input was validated on the user end and the function calls to retrieve the right entries from database was a success
2. Second use case in regards to system response time was a success with one or more computers being used to run the system
3. Third use case in regards to security was a success. The two users, Administrator and User, each had the appropriate data presented to them based on their role and the abilities for what they can do on the system was also separated based on role type.
4. The fourth use case was successfully without failure being testing in the environment the clients would be using
5. Fifth use case was successful as the database methods with retrieving, storing, and deleting entries work as intended