CEng 492

DOOR SECURITY PROJECT

by C4-CORP

TEST SPECIFICATION REPORT

1190545 Özkan Kılıç
1248715 Kerim Şahin
1255603 Okan Bozkurt
1255637 M.Bilal Demirkan

Table of Contents

Test Specification Report

C4-CORP

Project members: 1190545 Özkan Kılıç
1248715 Kerim Şahin
1255603 Okan Bozkurt
1255637 Mustafa Bilal Demirkan

Executive Summary

Test Specification Report is written for the project that is the production of a card based wireless door security system. It is the phase to make testing steps definite. First, the introduction is given to provide reader with the aim of the report. Second, test design specification and test environment is given to determine the test approaches in specific settings. Third, test case specification is given to declare test cases to each module. Finally, test procedure specification is to provide a sequence of steps to analyze and test the modules. The specifications will provide testers with necessary information throughout the debugging, quality improvement, validation and revision processes.

1 Introduction

This document will define and outline the test methodologies and test specifications for Wireless Door Security Project developed by C4Corp Team.

Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to project quality and widely deployed by programmers and testers, testing still remains an art, due to limited understanding of the principles of software. The difficulty in testing stems from the complexity of system: we can not completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, reliability estimation, a generic metric, and trade-off between budget, time and quality.

The report consists of test methodologies of modules in terms of functionality. Testing against the Software Requirement Specification is a validation process to ensure that the delivered system has met the users' specified refinements and needs as we identified before.

2 Test Design Specification

In this section, we will mention about the test approaches we have planned to apply to our project. In our detailed design report, we clarified the modules of the project as follow:

1. Hardware
   - Card design
   - Connections
2. Software
   - Database and
   - Administrator program
   - Main Board Program

In order to get benefit from the modularity, we will apply module (unit) testing to the modules. Module Testing is applied at the level of each module of the system under test. Module testing is usually done by the developers themselves, before the module is handed over for integration with other modules. Usually the testing at this level is structural (white-box) testing. Module Testing is sometimes called Unit Testing representing the lowest level component available for testing. We will test each module to determine its functionality. However, we will have different testing approaches to different module because our project is a composition of hardware and software modules. Note in some situations, unit testing is also used to refer to testing of logical or functional units that have integrated a number of modules. We consider this to be integration testing.

After testing the modules of our door security project, we will apply integration testing. First, integration testing of Administrator Program and Database is explained. Then, we apply integration testing to Main Board Program, C4 Control Card, and Database. Integration Testing is an interim level of testing applied between module testing and system testing to test the interaction and consistency of an integrated subsystem. Integration testing is applied incrementally as modules are assembled into larger subsystems. Testing is applied to subsystems which are assembled either:

- ✓ Bottom-up - assembles up from the lowest level modules and replaces higher level modules with test harnesses to drive the units under test.
- ✓ Top-down – assembles down from the highest level modules replacing lower level modules with test stubs to simulate interfaces
- ✓ Mixture of both bottom-up and top-down.

Integration testing is performed so a partial system level test can be conducted without having to wait for all the components to be available, and thus problems can be fixed before affecting downstream development on other components. Thus, integration testing is usually done as soon as identifiable logical subsystems are complete.

For example, we apply integration testing to the unification of Administrator Program and Database. If there occurs any problem, let's say, Administrator Program cannot update the Database, then we will work on fixation of this problem as a software module. If we are unable to detect this problem, then we will have to work on it after the unification of hardware and software modules. Unfortunately, it will be very difficult to fix the same problem because we will hardly be able to detect whether there is a hardware or software communication problem or not.

Integration testing is also performed as some tests cannot be carried out on a fully integrated system, and requires integration with special test harnesses and stubs to properly test a component. It also helps to focus testing to particular components and to isolate the problems that will be discovered. Usually the testing at this level is a mixture of functional (black-box) and structural (white-box) testing.

For example, for our project, assume the modules were tested and they functioned as they had been expected. Then, the card and connections will be integrated and tested. Afterwards, the communications between administrator and main board program will be tested following the integration of wireless communication. Then, the database module will be integrated to these modules and they will be tested all together. This

approach will guarantee that the total bugs in the system will be lessened before the system testing.

Finally, we will perform system testing. System testing represents testing after a complete system has been assembled. System testing is usually performed after all modules, integration and unit testing have been successfully applied. We will test the complete system, with possible scenarios to see the functionality. Our test cases will be created in such a way that they will represent the expected working principles of the whole system from general to specific.

Unfortunately, it is inevitable to miss some cases and bugs during the testing. Versioning is a solution to this obligation.

To summarize our approaches:

- Module Testing:
  - Testing of modules (Card, Connections, Database, Administrator program, Main Board Program)
- Integration Testing:
  - Testing of functionality of integrated modules incrementally.
    - Administrator Program and Database Integration.
    - Main Board Program, C4 Control Card and Database Integration.
- System Testing:,
  - Testing of whole system with test cases that cover the usage from general to specific.

2.1 Testing Environment

The test cases will be performed in the following configuration

- VIA ITX Main Board (with 2xUSB + 1xSerial + 1xPCI + 1xParallel port) with a monitor and keyboard connected
- Power supply for Main Board
- 128MB-memory
- 128MB-Disk space
- Wi-Fi Ethernet Card

- 2 Wiegand Proximity Readers
- Door Status Sensor
- Lock
- C4 Control Card
- A laptop (ASUS DC2000, AMD2800) to run administrator module.

3 Test Case Specification

3.1 Module Testing Cases:

This section is related to the explanation of test cases for each module: Administrator Program, Database, Main Board Program, Connection modules.

3.1.1 Administrator Program (AP):

Test#AP 1: Testing the links on the administrator program. (Black-box Test)

3.1.2 Databases:

Test#DB1: Testing data inputting. (Block-box Test)
Test#DB2: Testing data editing and querying. (Block-box Test)

3.1.3 Main Board Program (MP):

Test#MB 1: Testing data packaging.

3.1.4 Connections:

Test#C 1: Testing the wireless connection and communication.

3.2 Integration Testing Cases:

This section is related to the explanation of test cases for integration of modules incrementally.

3.2.1 Administrator Program and Database Integration

Test#APD 1: Testing the login to the AP. (Security Test)
Test#APD 2: Testing the user management features of AP.
Test#APD 3: Testing the card management features of AP.
Test#APD 4: Testing the door management features of AP.
Test#APD 5: Testing the scheduling features of AP.

3.2.2 Main board Program, Database and C4 Control Card Integration

Test#MDC 1: Testing the booting the MP.

Test#MDC 2: Testing the registered card reading and granting access.

Test#MDC 3: Testing the unregistered card reading and signaling.

Test#MDC 4: Testing the data inputting from MP to Database.

3.3 System Testing Cases:

This section is related to the explanation of test cases for integration of modules incrementally.

Test#ST 1: Testing starting up the system.

Test#ST 2: Testing the real-time monitoring.

Test#ST 3: Testing the direct door controlling.

Test#ST 4: Testing the MB database updating initiated by AP.

Test#ST 5: Testing the UPS.

Test#ST 6: Testing the log transfer from MB to AP.

Test#ST 7: Testing the system performance.

Test#ST 8: Testing the overload.

4 Test Procedure Specification

Test procedure specification is to provide a sequence of steps to analyze and test the modules, integrations and whole system. The steps of each test case are as follows:

Test#AP 1:

1. Administrator program is run in a web browser on the master computer.
2. Main menu is displayed.
3. All the buttons and links are clicked consecutively. Corresponding new menus and screens are displayed.

Test#DB 1:

1. MySQL Server is run on the master computer.
2. The following SQL statement for table creation is executed:

```
CREATE TABLE tblDepartments (
fldDEPDepartment_No INT UNSIGNED PRIMARY KEY NOT NULL,
fldDEPDepartment VARCHAR(30) NOT NULL
            );
```

3. The table creation is checked.
4. The following SQL statement for inserting data is executed:

```
INSERT INTO tblDepartments VALUES (15,'Computer
Engineering');
```

5. The inserted value is checked by following SQL statement:

```
SELECT * FROM tblDepartments;
```

Test#DB 2:

1. MySQL Server is run on the master computer.
2. The following SQL statement is run to update data:

```
UPDATE tblDepartments SET fldDEPDepartment='Civil
Engineering'
WHERE fldDEPDepartment_No=15;
```

3. Updated data is checked by the following SQL statement:

```
SELECT * FROM tblDepartments;
```

Test#MB 1:

1. Code is modified so that the prepared string package will be printed to the standard output temporarily. Furthermore, the code is modified so that main board program sends a log report automatically.
2. The modified code is compiled and run.
3. It is observed that the format of printed string (data package to be sent) is same as it is indicated in detailed design report:

DoorID	MessageTypeCode	DataLength	Data

Test#C 1:

1. A keyboard is connected to the main board.
2. Boot the board.
3. Wireless connection is checked by executing the following command in the shell prompt:
   hostname –i ifconfig
4. Wireless communication between the main board and administrator computer is checked by executing the following command in the shell prompt:
   ping <*admin-ip*>

Test#APD 1:

1. Administrator program and MySQL Server are run in a web browser on the master computer.
2. In the login screen, user name: “demirk” and password: “c4corp” are entered.
3. Main menu is displayed successfully.

Test# APD 2:

1. Administrator program and MySQL Server are run in a web browser on the master computer.
2. Administrator program is logged into.

3. User Management button is clicked.
4. New user is added to the system by entering necessary data into provided spaces: e.g. user name “Hakan”
5. “Hakan” is checked from the user list.
6. “Hakan” is edited into “Serkan” and “Serkan” is checked from the user list.

Test# APD 3:

1. Administrator program and MySQL Server are run in a web browser on the master computer.
2. Administrator program is logged into.
3. Card Management button is clicked.
4. New card is added to the system by entering necessary data into provided spaces: e.g. Card id “0123456789”
5. “0123456789” is checked from the card list.
6. Newly added card with the id “0123456789” is deleted by choosing and deleting it from the card list.

Test# APD 4:

1. Administrator program and MySQL Server are run in a web browser on the master computer.
2. Administrator program is logged into.
3. Door Management button is clicked.
4. New door is added to the system by entering necessary data into provided spaces: e.g. door id “15”
5. “15” is checked from the card list.
6. Newly added door with the id “15” is deleted by choosing and deleting it from the door list.

Test# APD 5:

1. Administrator program and MySQL Server are run in a web browser on the master computer.
2. Administrator program is logged into.
3. Scheduling button is clicked.

4. A new schedule for a private laboratory that can only be entered with appointment is arranged to a closer time (e.g. time=11:00-11:30, access group=2) by entering the necessary data into the provided fields and saved.
5. A card from access group=2 is brought closer to the card reader of the private laboratory at 10:55.
6. The access is denied.
7. A card from access group=2 is brought closer to the card reader of the private laboratory at 11:10.
8. The access is granted.
9. A card from access group=2 is brought closer to the card reader of the private laboratory at 11:50.
10. The access is denied.

Test#MDC 1:

1. Power button is pressed.
2. The operating system is booted and the operating system is loaded on the screen.
3. MySQL Server, C programs (client.exe, server.exe, common.exe) and wireless connection are observed to be active.

Test#MDC 2:

1. A registered card is brought closer to the card reader after the boot.
2. The lock is opened and green led is lighted and beeper is activated.

Test#MDC 3:

1. An unregistered card is brought closer to the card reader after the boot.
2. The lock is not opened and red led is lighted and beeper is activated.

Test#MDC 4:

1. A registered card is brought closer to the card reader after the boot.
2. MySQL prompt is activated.

3. Log record of recently read card is observed in the database by querying from the log table.

Test#ST 1:

1. Main computer and main boards are booted.
2. All the programs, Administrator, MySQL Server, C programs (client.exe, server.exe, common.exe) and wireless connection, are checked to be run. Led is yellow.
3. Database update time of each main board is verified by checking the modification time of “cards” file.

Test#ST 2:

1. The system is booted.
2. Real time monitoring feature of administrator program is enabled by clicking to the monitoring button.
3. A record is displayed at each time a card is read by the card reader at the monitoring screen of the AP.

Test#ST 3:

1. The system is booted.
2. Access Management button at administrator program’s main menu is clicked.
3. Lock all the door command is given.
4. A registered card is brought closer to the card reader.
5. The lock is observed to be still locked.
6. Unlock all the door command is given at Access Management.
7. The locks are observed to be unlocked.

Test#ST 4:

1. A change is made in the database of the AP.
2. A database update is made, for example, by adding a new card.
3. Database update time of each main board is verified by checking the modification time of “cards” file.
4. A latency that is very close to the current time is observed.

Test#ST 5:

1. The system is booted.
2. The electricity is cut manually.
3. It is observed that the system does not shut down and all programs keep running, i.e. Administrator, MySQL Server, C programs (client.exe, server.exe, and common.exe) and wireless connection, are observed to be active because of UPS.

Test#ST 6:

1. A card is read 100 times (Event capacity of a door is 100 which indicates that MB program sends the log file to the admin program when the number of records in tblLogs reaches 100).
2. 100th card's reading time is noted down.
3. MySQL prompt is activated at master computer.
4. Records of tblLogs are queried by following SQL statement:

```
SELECT * FROM tblLogs;
```

5. It is observed that last record's creation time is same with the time noted at the step 2.

Test#ST 7:

1. A change is made in the database of the AP.
2. A database update is made, for example, by adding a new card.
3. Current time is noted down.
4. Database update time of each main board is verified by checking the modification time of "cards" file.
5. This modification time is compared with the time noted down at step 3.
6. The time difference is observed to be less than 5 seconds.
7. A registered card is brought closer to the card reader.
8. It is observed that the lock is unlocked in less than 1 second.

Test#ST 8:

1. All the main boards are booted at the same time so that all the database update requests are received at the same time by the master computer.
2. The databases of all main boards are observed to be updated accurately.
3. Monitoring button at administrator program's main menu is clicked.
4. Real time monitoring of all doors are enabled by administrator computer.
5. 10 cards are brought closer to 10 doors' card readers at the same time.
6. It is observed that all of 1o cards' information is monitored in the administrator program accurately.

5 Test Revisions

Test specification steps should be carried out as we stated in test procedure specification section. During the execution of test cases, the system should respond as it is specified in the steps of each test. In case of unexpected behavior, crush, dead lock, output, and such, note down the responses carefully. Afterwards, the production team will revise the system to eliminate the errors. After the revisions, the test case with erroneous response will be performed again until it results as expected.
This test specification report is a living document. New test cases can be added later and new revisions will be done with respect to new ones to get a well functioning final product of the whole system.