10.0 Enterprise Wireless Connectivity

10.1 Enterprise Wireless Connectivity

All businesses have the increasing need to change their enterprise networks in order to accommodate the rapidly growing demand for wireless connectivity. Many enterprises, such as Humber College, are now compelled to build their own enterprise wireless network to support the influx of mobile devices, such as, Internet of Things, and cloud-based applications, as well as increased adoption of WiFi-hungry practices like Bring Your Own Device (BYOD). An enterprise-grade wireless network is more than just a collection of WiFi Access Points (APs). At minimum, it's characterized by superior security and performance; centralized configuration and management; and a much higher capacity for user density. A wireless network is very similar to a wired network with one big difference: Devices don't use cables to connect to the router and one another. Instead, they use radio wireless connections called Wi-Fi (Wireless Fidelity), which is the 802.11 networking standards supported by the Institute of Electrical and Electronics Engineers (IEEE).

In regards to the MAIDS project, wireless connectivity was achieved using the Humber College wireless network with the following set parameters:

a. Range: Up to 200 m
b. Frequency: 2.4 GHz 5 GHz
c. PHY Throughput: Up to 72 Mbps
d. Network type Peer-to-peer: Star

The Humber College wireless network security is implemented using Wi-Fi Protected Access (WPA2) encryption and multiple layer of security like TLS. In addition, Network address translation (NAT) is a method of remapping one IP address space into another. With NAT, one public IP address can hide a number of private IP addresses. Finally it employs Quality of Service (QoS). QoS provides the ability to prioritize different applications, users, or data flows to guarantee a certain level of performance. The actual code used in the wpa_supplicant file to connect to the Humber College wireless network is found in Appendix II of this report.

10.2 MAIDS Database Accessibility: Prototype and Mobile Application

10.2.1 LAMP Installation and Configuration

In order to access the MAIDS database, it must first be installed and configured. MySQL installation and configuration requires a series of prior steps to install it using the LAMP stack. MySQL is a popular relational database system (RDBS). A relational database stores data in a structured format, using rows and columns (Christensen, 2017) and is commonly included in the LAMP (an acronym of the names of the original four open-source components: Linux, Apache, MYSQL, and PHP) stack. MYSQL is a key component of dynamic websites and the best way to store data for web applications. MySQL allows the user to store and maintain large amounts of data easily.

10.2.2 Minimum MySQL Hardware Requirements

The minimum hardware requirement to run the MAIDS' MySQL database include the following:

a. CPU: Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD CPU
b. Cores: Single (Dual/Quad Core is recommended)
c. RAM: 4 GB (6 GB recommended)
d. Graphic Accelerators: nVidia or ATI with support of OpenGL 1.5 or higher
e. Display Resolution: 1280 × 1024 is recommended, 1024×768 is minimum.

10.2.3 LAMP Stack Installation

Installing the LAMP stack requires the procedure outlined in the following sections.

10.2.3.1 Update the Raspbian Operating System

1. Update the Raspbian OS executing the following commands: sudo apt-get update and sudo apt-get upgrade –y.

10.2.3.2 Install Apache2 Server

To install Apache2 Server execute the following steps and command sequence:

1. Install the Apache2 server and restart it executing the following steps and commands: sudo apt-get install apache2 –y and sudo systemctl restart apache2.
2. Once apache2 is installed execute ifconfig eth0 command to display the IP address of the Raspberry Pi.
3. Then, enter the IP address into the browser's address bar and if installed properly, the user should be presented with the Apache2 Debian Default Page.

10.2.3.3 Install PHP Package

To install PHP execute the following steps and command sequence:

1. Install PHP package and dependencies
    a. sudo apt-get install php libapache2-mod-php –y
2. Create an index.php file with the following content: <?php echo "hello world"; ?>.
3. Using the web browser, enter into the browser's bar the following: http://xxx.xxx.xxx.xxx/index.php
4. A correctly installed PHP package will display the following message: "Hello World".

10.2.3.4 Install MySQL Database (mariadb)

To install the MySQL database execute the following steps and command sequence:

1. Install the maridb database and dependencies: sudo apt-get install mariadb-server mariadb-client php-mysql –y
2. Then, restart the Apache2 server executing the command: sudo systemctl restart apache2
3. Finally, secure the MySQL installation with the command: mysql_secure_installation

At this point, the LAMP installation is finished.

10.3 MySQL Database Password Configuration Procedure

Once installed, MySQL has no root password and it won't work with MariaDB unless one is specified. Configuring MySQL and assigning a password to the root user requires the following procedure:

1. Start the MySQL database: sudo mysql -u root
2. Enter root user password (specified during MySQL installation)
3. reset the root password with the following statement:
    a. ALTER USER 'root'@'localhost' IDENTIFIED BY 'xxxxxxxxxxxxxxx';
4. Quit MySQL and login again to create the MAIDS database.
5. Follow steps 1 and 2 (with new root password)

10.4 Create maids1 Database and maidsintrusion Table

1. Execute the following command to create a database:
    a. `CREATE DATABASE maids1;`
2. Choose the maids 1 database executing the command:
    a. `use maids1;`
3. Next, create the table ‘maidsintrusion’ with the CREATE TABLE statement as follows:

```
CREATE TABLE `maidsintrusion` (
# id of record
  `id` int() NOT NULL AUTO_INCREMENT,
# intrusion address
  `address` varchar(25) NOT NULL,
# intrusion location
  `location` varchar(25) NOT NULL,
# intrusion date
  `intrusiondate` varchar(30) NOT NULL,
# person’s name reporting intrusion to authorities
  `reportingperson` varchar(25) NOT NULL,
  # reporting person’s contact phone
  `contactphone` varchar(25) NOT NULL,
  # reporting person’s contact email
  `contactemail` varchar(25) NOT NULL,
    PRIMARY KEY (id)      # Make the id the primary key
);
```

10.5 Establishing Database Connectivity

Accessibility to the MAIDS database was accomplished as follows:

1. The Python3 dbinsert() function imports the mysql.connector and Error modules to handle the connection to the database and any errors that might result during the connection try. (i.e. import mysql.connector and from mysql.connector import Error)
2. Then, connection parameters are defined as follows:
    a. Host Ip Address (host=192.168.xxx.xxx)
    b. Database Name (database=maids1)
    c. Username (dbusername=clauxxxxxxxx)
    d. Database Password (dbpassword=xxxxxxxxxx)
3. Finally, the function configures a connection to the database server on MAIDS by using the following command:
    a. connection = mysql.connector.connect(host,database,user,password)
4. A message is displayed on the Android application if the connection is successful or not.

10.6 Inserting Data into Database

Once the connection to the database is made, the data from the intrusion is gathered with the following statement into the db_data variable:

a. db_data = (address, location, intrusiondate, reportingperson, contactphone, contactemail)

The data form the intrusion is then formatted for insertion into the database using a parametrized insert statement:

a. mySql_insert_query = """INSERT INTO maidsintrusion (address, location, intrusiondate, reportingperson, contactphone, contactemail)VALUES (%s, %s, %s, %s, %s, %s) """

Once the parametrized statement is constructed, the data is inserted into the database with the following command:

a. cursor.execute(mySql_insert_query, db_data)

Finally, the information is committed to the database and the connection is closed with the following commands:

a. connection.commit()
b. print("Record inserted successfully into MAIDS-DB")
c. cursor.close()
d. connection.closed()

10.7 Testing Database Connectivity

Testing connectivity to the MAIDS' MySQL database employed a simple TCP connection (Telnet) using putty software. Putty is a free and open-source terminal emulator application that supports several network protocols (i.e. SCP, SSH, Telnet, rlogin, and raw socket connection). The advantage of using telnet is that is extremely simple. There are no configuration files to modify and no authentication. It either makes the connection or does not. The database was accessible. Therefore, further successful testing was done with the MAIDS' Android application.

10.8 Security Considerations

Risks are inherent in any application requiring users to supply input (i.e. usernames, passwords, personal information, etc.) Database attacks resulting in data breaches may come from SQL virus attacks or from employee misuse. Successful attacks can lead to the theft of thousands (sometimes millions) of records containing valuable data (i.e. personal information, credit card, financial, healthcare, etc.) Therefore, the main purpose behind any cyber-attack is to get access to a database located on a server and it is therefore imperative that the security of the database server is strengthened. The strengthening of the database depends on database and network security, operating system hardening and physical security. The MAIDS server database was secured in the following ways:

1. Deploying a strong password policy execution – A strong password policy is the front line of defense to confidential user information. A password policy is a set of rules which are created to improve computer security by motivating users to create dependable, secure passwords and then store and utilize them

properly. MAIDS implements a strong password policy based on the following criteria:

a. At least 8 characters long
b. Should not contain personal information (i.e. username, person name, etc.)
c. Should not contain dictionary words.
d. Should contain uppercase letters, lowercase letters, numbers, and characters

2. Discard Default Users and Demo-test Databases – Default users and demo databases are public record. Therefore, MAIDS removed default users and demo databases so that an attacker is prevented from collecting details on the database and user information.
3. Implement MySQL Enterprise Firewall – MySQL Enterprise Firewall guards against cyber security threats by providing real-time protection against database specific attacks. MAIDS implements the MySQL Enterprise Firewall to protect data by monitoring, alerting, and blocking unauthorized database activity without any changes to the application. The system variable mysql_firewall_mode was set to ON in order to implement the MySQL Enterprise Firewall (i.e. mysql_firewall_mode=ON)
4. Changed the Admin User Name – For example, the name "admin" is not very secure, because it's easier to hack via brute force. Therefore, for additional security, MAIDS changed the admin username.

5. Restricted User Privileges – Access and privileges were limited to the database user's needs. This will help in preventing data loss even after an exploit attempt.
6. Disabled Public Network Access to the Database Server – All public network access to the database server was blocked.
7. Implemented iptables and Malware Scanners – iptables (Linux-based firewall) was setup on the operating system and malware scanners were frequently ran on the database server.
8. Changed the port for the outside – Changing the database default port (3306) will stop bots that target said port from directly attacking that specific port.
9. Set MySQL SSL – SSL stops transmitting data in clear between the client and the server.
10. Only allowed access from certain IPs -- This ensures that only certain IP's will be able to connect to the database server (using iptables)

10.9 Unit Testing

Unit testing is a software testing method by which individual units of code are tested in isolation. The purpose of unit testing is to isolate the smallest testable parts of an API and verify that they function properly in isolation. A unit test can verify different behavioral aspects of the system under test (SUT), but mainly it verifies that the SUT produces the correct results. Unit testing ensures the whole application or system works as expected by ensuring the critical application and device capabilities, such as user logins, network connections, database reachability, Internet services connections, and all the critical hardware behaves as expected.

The benefits of writing and performing unit-tests are as follows:

a. Makes the coding process more Agile.
b. Issues can be found very early and can be resolved.
c. Refactor code or upgrade system libraries and make sure modules work correctly.
d. Provides documentation of the system.
e. Simplifies the debugging process.
f. Helps reduce the cost of bug fixes.
g. Increases confidence in changing/ maintaining code.
h. Codes are more reusable.

The table below lists the unit testing results for the MAIDS device.

TABLE 1 UNIT TESTING RESULTS FOR MAIDS PROJECT.

Component Tested (SUT)	Expected Behavior	Actual Test	Expected Result
Motion Module	Motion is sensed when object moves in front of the motion module	a. Person moves in front of MAIDS motion sensor module b. Motion test Python3 code is ran	Motion is sensed and appropriate response is triggered.
Sound Module	Sound is sensed when object moves in front of the sound module	a. Person recreates a loud sound in front of MAIDS sound sensor module b. Sound test Python3 code is ran	Sound is sensed and appropriate response is triggered.
LED Module	LED module lights up green or red depending on sensing situation	a. Motion/sound sensors are not triggered. b. Motion/Sound sensors are triggered	d. Motion/sound sensors triggered LED module lights up green.

		c. Run LED module Python3 code.	e. Motion/sound sensors triggered LED module lights up red. f. Module runs and green/red LED's light up.
Camera Module (No python3 code was ran)	Camera module triggered by room intrusion	Trigger sensors (motion/sound) one-at-a-time to trigger the camera module	Motion/sound sensors triggered and photograph is produced.
Switch Module (No Python3 code was ran)	Switch module powers MAIDS device when switched ON and powers MAIDS device down when switched OFF	a. Press switch to ON position b. Press switch to OFF position	c. MAIDS' device powers up. d. MAIDS' device powers down.
Database Module	Database reached via Internet returns	"Retrieve database data" and "Retrieve intrusion photo"	a. Database data is retrieved and displayed on

	recorded data and intrusion picture	buttons clicked in Android software activity (Python3 code ran when button clicked)	Android application screen. b. Intrusion photo is retrieved and displayed on Android application screen.
Enterprise Network Connection	Connect to any enterprise network defined in wpa-supplicant file	a. Connect to home network b. Connect to Humber College Network c. Connect to Tim Hortons' network	Connection to all networks defined in wpa-supplicant file achieved.
Internet Services Connection (Twilio, PushNote, eMail, phone call)	a. Receive Twilio SMS message when sensors triggered. b. Receive PushNote	a. Trigger sensors to receive SMS message. b. Trigger sensors to receive	When sensors were triggered the following occurred: a. SMS message received.

	message when sensors triggered. c. Receive email message when sensors triggered. d. Receive phone call when sensors triggered.	PushNote message. c. Trigger sensors to receive email message. d. Trigger sensors to receive phone call.	b. PushNote message received. c. Email message received. d. Phone call received.
Android Application Login	User inputs username and password and log into the Android application	Input the username and password into Android application Login Activity	Entered username and password allows login into Android application
Control activity – Activate MAIDS	Remotely activate MAIDS device	Click on “Activate MAIDS” button in Android application	MAIDS device is remotely activated
Control Activity – Deactivate MAIDS	Remotely deactivate MAIDS device	Click on “Deactivate MAIDS” button in Android application	MAIDS device is remotely deactivated

10.10 Production testing

In production testing, the tester is not concerned with the executable code. Instead, the tester is concerned with the need to verify the output of the MAIDS device based on given user requirements against the expected output. The prime objective of MAIDS production testing is to check the functionalities of the system. These tests, check the Android application, device hardware (Raspberry Pi 4 and custom-made PCB board), and networking infrastructure, from the front end UI to the back-end database systems. In that sense, the MAIDS production tests are also a form of integration testing, ensuring that different components are working together as expected.

The MAIDS' production testing environment was designed to measure and monitor the following:

1. Measure the application's performance in real-time.
2. Monitor the application in real-time to detect network failure and weak connections.
3. Monitor the API responses at peak traffic.
4. Detect bugs which typically go unnoticed.
5. Detect possible point(s) of failure.
6. Help maintain the high quality of the device and application.
7. Produce a reliability statistic.

Consequently, individual, as well as an integrative, production tests were performed and the results tabulated below.

TABLE 2 PRODUCTION TESTING RESULTS FOR MAIDS PROJECT.

Test No.	Components tested	Expected Results	No. of Trials	Success Rate (%)	Failure Rate (%)	Actual Results
1	PCB board, sound sensor, LED module, audible messages, network connection, Android application	Android application remotely triggers PCB board and sensor module exchange intrusion information that triggers the LED module and audible messages.	100	98	2	PCB board, sound sensor, LED module and audible, network connection and Android application performed as expected.
2	PCB board, motion sensor, LED module, audible messages,	Android application remotely triggers PCB board and sensor	100	99	1	PCB board, motion sensor, LED module, audible, network connection and

	network connection, Android application	module exchange intrusion information that triggers the LED module and audible messages.				Android application performed as expected.
3	PCB board, LED module, network connection, Android application	Android application remotely triggers PCB board and LED module information exchange that triggers the LED module (green or red)	100	100	0	PCB board, LED module, network connection and Android application performed as expected.
4	PCB board, network connection,	Android application triggers PCB	100	100	0	PCB board, network connection and

	Android application (MAIDS remote activation)	board to activate MAIDS device.				Android application performed as expected.
5	PCB board, network connection, Android application (MAIDS remote deactivation)	Android application triggers PCB board to deactivate MAIDS device.	100	100	0	PCB board, network connection and Android application performed as expected.
6	PCB board, motion sensor, network connection, Twilio services (SMS message and phone call).	PCB board and motion sensor module exchange intrusion information that triggers the LED module,	100	100	0	PCB board, motion sensor, network connection and Twilio service (SMS and Phone call) performed as expected.

		audible message, and Twilio service.				
7	PCB board, motion sensor, network connection, PushNote service (mobile pushnote).	PCB board and motion sensor module exchange intrusion information that triggers the LED module, audible message, and PushNote service.	100	100	0	PCB board, motion sensor, network connection and PushNote service (mobile pushnote) performed as expected.
8	PCB board, motion sensor, network connection,	PCB board and motion sensor module exchange intrusion	100	100	0	PCB board, motion sensor, network connection, email service and camera

email service and camera.	information that triggers the LED module, audible message, and emails intrusion photo and data to recipient.		performed as expected.