Unsolvable Solutions
Client: Francois Mouton at the CSIR DSSR

Functional Requirements

Eavesdropping Protection in Conclave

Github link: `https://github.com/Unsolvable-Solutions/Project-EPIC`

Members:	*Student Number:*
Edwin Fullard	12048675
Jaco Bezuidenhoudt	11013878
Jandre Coetzee	10693077
Maret Stoffberg	11071762
Ryno Pierce	12003922

Contents

1 Introduction

1.1 Project Background

The Android Operating System officially took over the smart phone market in 2010 and it is suspected that about 700 000 Android devices are used in South Africa. It is mostly the corporate or more upper class communities that use these smart phone devices. It is also these individuals who sit in the big corporate meetings where extremely sensitive data can be discussed. For this reason, if these individuals could have eavesdropping malware on their smart phone, it could cause sensitive data to be easily leaked out.

1.2 Project Vision

The Eavesdropping Protection in Conclave (EPIC) aims to protect the integrity of the information discussed during a meeting by eliminating access to the mobile device during a meeting.

1.3 Project Scope

The Eavesdropping Protection in Conclave(EPIC) product consist of a mobile application, a web page, a server, a gateway and a node.

- A meeting is scheduled via the web page. On the creation of the meeting people are invited via email.
- Before the meeting is held, the gateway and node is set up with the list of invited people that may access the meeting room.
- At the meeting, the person opens the application on his phone. He then holds the phone over the node. The node then replies if the person has permission to access the room or not. If the person has permission the meeting room is unlocked.
- If access is denied, the gateway sends a refresh request to the server, and the person can try again to gain access.
- When permission is granted, the application will start the protecting mode on the phone.
- The server keeps a log of all the activities. This log can be queried afterwards by the creator of the meeting.

2 Functional Requirements

2.1 Server

2.1.1 Scope:

- The server is used with the gateway: it sends and receives data to and from the server.
- A user can add, remove and edit meetings on the server via the website.
- A user can add people to a meeting and remove people from a meeting on the server via the website.
- The server stores a database with all the data.

2.1.2 Functionality

2.1.2.1 getMeeting

Priority: Critical

Service Contract: A query is made for a specific meeting and the function returns a JSON object with the meeting that meet the query.

Pre-conditions:

- The query must be valid.
- The meeting must exist.

Post-conditions:

- A meeting object that meet the query is returned.
- The action is logged on the server.

2.1.2.2 getPeopleList

Priority: Critical

Service Contract: This function returns a list of Person object that may attend a specified meeting.

Pre-conditions:

- The meeting must be specified.
- The meeting must exist.
- Only the owner of the meeting or the gateway may call this function.
- The server must be running.

Post-conditions:

- A list of Person objects is returned.
- The action is logged.

2.1.2.3 Create Person

Priority: Critical

Service Contract: This function is called when a person is added to a meetings people list and the person is not yet in the database. The function creates a Person object and stores it in the database.

Pre-conditions:

- The user must have permission to add the person.
- The required fields are email address, name and surname.

Post-conditions:

- The person object is created and it is added to the database.
- The action is logged in the database.

2.1.2.4 Create meeting

Priority: Critical

Service Contract: The function is called by the website when the user creates a meeting. The meeting object is then created and stored in the database.

Pre-conditions:

- The meeting title, meeting room, date and time are required fields.

Post-conditions:

- The meeting object is created and it is added to the database.
- The action is logged in the database.

2.2 Website

2.2.1 Scope

The website is used by a user to manage his meetings.

2.2.2 Functionality

2.2.2.1 Register

Priority: Important

Service Contract: The user enters his name, surname, email address and password. These details are stored in the database on the server and the user is automatically logged in.

Pre-conditions:

- The user must be connected to the Internet.
- The server must running.

Post-conditions:

- The users name, surname, email address and password are stored in the database.
- login function is called with the users details.
- The action is logged in the database.

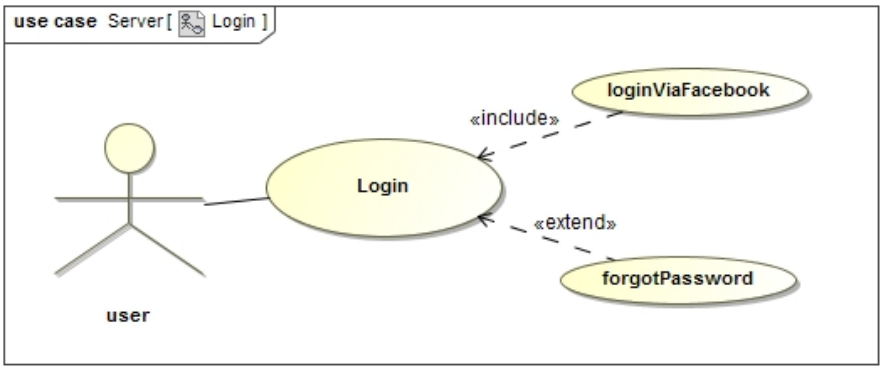

Figure 1: A Use Case Diagram of the Login services

2.2.2.2 Log in

Priority: Important

Service Contract:The user types his email address and password to enter the website

Pre-conditions:

- The browser must be connected to the Internet.

- The server must running.
- The user must be registered.
- The password must be correct.

Post-conditions:

- The user becomes the current user on the website session.
- The user is on his home page.
- The action is logged in the database.

2.2.2.3 Log out

Priority: Important

Service Contract: The user clicks on a button and the user is logged out of his account.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.

Post-conditions:

- The user is logged out.
- The current user on the session becomes null.
- The browser is redirected to the log in page
- The action is logged in the database.

2.2.2.4 Reset password

Priority: Nice to Have

Service Contract: The user can reset a forgotten password. The function is available on the log in page.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user should not be logged in.
- The user has to provide an email address.

Post-conditions:

- An email is send to the users email address with a link to the reset password web page. The user has to open it.
- On the reset password page, the user enters the password twice.
- The users password is changed in the database.
- The user is redirected to the log in page.
- The action is logged in the database.

2.2.2.5 Create Meeting

Priority: Critical

Service Contract: The user schedules a meeting by entering the title, description, meeting room, time and date.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.
- The meeting title, meeting room, date and time are required fields.

Post-conditions:

- The meeting and its details are stored in the database.
- The meeting is added to the users list of meetings on the database.
- The update meeting, invite person and remove meeting functions are available.
- The action is logged in the database.

2.2.2.6 Invite Person

Priority: Critical

Service Contract: The user invites a person to a specific meeting. This function is called for every person being invited.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.
- The meeting has to be specified.
- The required fields for the person being invited are: the name, surname and email address.

Post-conditions:

- If the person does not yet exist in the database he is added to the database.
- The person is added to the meetings list of invited people.
- An email is send to the person to notify him of the meeting.
- The action is logged in the database.

2.2.2.7 Remove Person

Priority: Important

Service Contract: The user removes a person from the list of invited people of a specified meeting.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.
- The meeting has to be specified.
- The person has to be specified.
- The person has to exist in the meetings list of invited people.
- The user has to confirm the action.

Post-conditions:

- The person is removed from the list of invited people of the meeting.
- The person is not removed from the database.
- An email is send to the person to notify him that he has been removed from the list.
- The action is logged in the database.

2.2.2.8 Update Meeting

Priority: Important

Service Contract: The user can update any of the specified meetings fields.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.
- The meeting has to be specified.

Post-conditions:

- The change is made to the meeting in the database.
- The action is logged in the database.

2.2.2.9 Remove Meeting

Priority: Important

Service Contract: The specified meeting is logically removed from the database.

Pre-conditions:

- The browser must be connected to the Internet.
- The server must running.
- The user has to be logged in.
- The meeting has to be specified.

Post-conditions:

- All the invited people are notified via email.
- The meeting is no on the owners list of meetings.
- The meeting is no longer on the invited peoples list of meetings.
- The action is logged in the databases.

2.3 Application

2.3.1 Scope

The Application is used when entering a meeting.

- The user will hold the mobile device over a node.
- The application will send the device identification and the persons email address to the gateway via the node.

  If permission is granted: The background of the application will turn green and the application will take a snapshot of the mobile device's current state and enable the protection. The snapshot is taken so that the mobile Device and application can be restored to its previous state after the meeting.

  If permission is denied: The background of the application will turn red and the application will do nothing.
- During the meeting the application check constantly if the protection mode is violated. If so, the protection mode is restored and the mobile device vibrates to alert the user of a possible attack.

- When exiting the meeting the user holds the mobile device over the node again and the mobile device is restored to its previous state.
- Overall system layout:

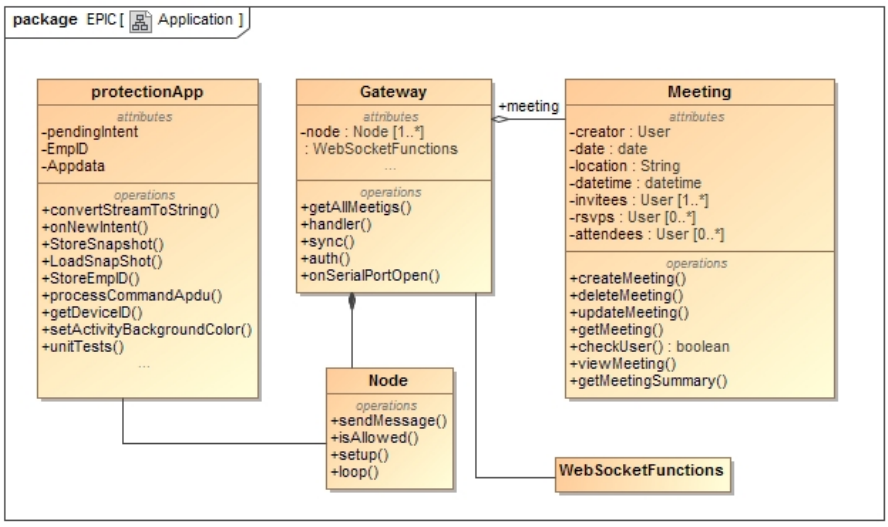

Figure 2: A Class Diagram of the protection Application, Meeting, Node and Gateway

2.3.2 Functionality

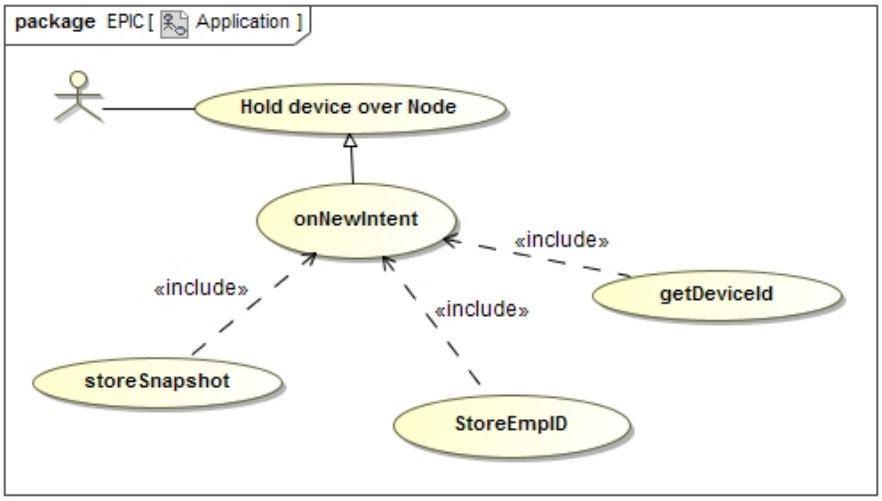

Figure 3: A Use Case Diagram protection Application

2.3.2.1 onNewIntent

Priority: Critical

Service Contract:This function is called when an Android system intent is triggered. It checks whether the intent was caused by NFC and then parses the NFC message that will check whether the user is allowed to enter the room. If the user is allowed, the store- and loadSnapshot functions are called respectively on whether the user is entering or leaving.

Pre-conditions:

- An NFC intent must be triggered by holding the mobile device against a node.
- Application either in safe or unsafe mode.

Post-conditions:

- According to the state of the application, either the LoadSnapshot or StoreSnapshot function should be called.

2.3.2.2 StoreSnapshot

Priority: Critical

Service Contract: This function stores the state of all communication mechanisms and then turns them off to enter *protection mode*.

Pre-conditions:

- The mobile device must have enough space to store the file with all the states.
- The mobile application must be in the unsafe state.

Post-conditions:

- A file is stored containing the current states of the mobile device's communication mechanisms.
- All communication mechanisms are turned off.
- The mobile application's state is switched to safe mode.

2.3.2.3 LoadSnapShot

Priority: Critical

Service Contract: This function is an overwritten function. It is used to create the user authentication data and return it when the mobile device is being read from via NFC.

Pre-conditions:

- The mobile application should be in the safe state.
- The connections should not have been toggled on while in safe state.

Post-conditions:

- The mobile application is switched to unsafe state.
- The connections of the mobile device are restored based on previous unsafe state.

2.3.2.4 StoreEmpID

Priority: Critical

Service Contract:This function stores a combination of fields entered by the user into a file that is used to with other functions to either retrieve the data if the application is closed and reopen, or send it via NFC to get use as authentication.

Pre-conditions:

- The Employee ID text field should not be empty.
- The ID needs to be valid.

- The mobile device needs to have enough space to store a file.

Post-conditions:

- A file containing the Employee ID is stored to the local mobile device.

2.3.2.5 getDeviceId

Priority: Important

Service Contract: Used to added an extra security layer to authentication by getting the unique device id.

Pre-conditions:

- The mobile device must be a valid device and have a unique id.

Post-conditions:

- The state of the mobile application should be unchanged.

2.3.2.6 unitTests

Priority: Critical

Service Contract:This function uses dummy scenario's to test whether all of the functions work as they should.

Pre-conditions:

- The mobile application must be in debugging mode.
- The mobile application must be in the unsafe state.
- The mobile application must have enough space to store the results.

Post-conditions:

- The mobile application should be in the unsafe state.
- A file with the unit test results is stored to the mobile device.

2.3.2.7 setActivityBackgroundColor

Priority: Nice to have

Service Contract: This function changes the background colour of the application to visually notify the user if he has been granted access to the meeting room.

Pre-conditions:

- The state of the application needs to be changed.

Post-conditions:

- The mobile application's background colour should be the new colour.

2.3.2.8 convertStreamToString

Priority: Nice to have

Service Contract: This function helps to convert an input stream of bytes into a *String type* that is used for file reading.

Pre-conditions:

- The input stream must be given

Post-conditions:

- The function returns a String object containing the value of the stream converted to a String data type.

2.4 Node

2.4.1 Scope:

- The node is connected to the gateway with a serial connection.
- The node uses NFC to communicate with the Protection App on the phone.
- When the phone is scanned, it sends the device id of the phone to the node. The node then sends it to the gateway to verify if the user has access to the meeting. When the gateway replies with the results of the verification, the node turns the lights the appropriate colour, decides if the door must be unlocked and sends feedback to the phone.

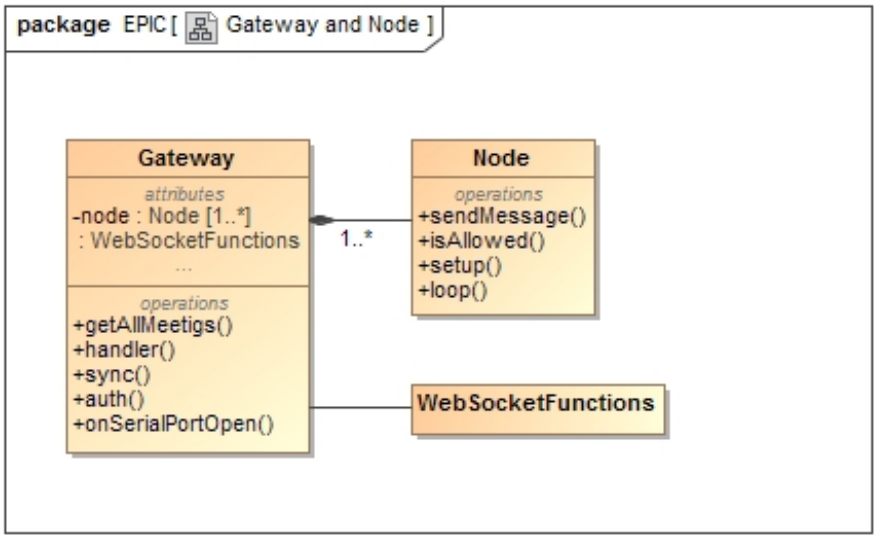

Figure 4: A Class Diagram of the Gateway and Node

2.4.2 Functionality

2.4.2.1 isAllowed

Priority: Critical

Service Contract: Sends the device ID to the Gateway and waits until it receives the results on whether or not to allow the user in.

Pre-conditions:

- The users' phone should be on the Node.
- The Protection App should have sent the device ID.

Post-conditions:

- The Gateway should have sent a reply.
- The sendMessage function should be called.

2.4.2.2 sendMessage

Priority: Critical

Service Contract: This function adapts the results to the correct format that the Protection App is expecting, and then broadcasts it via NFC to the phone. It also calls the setColor function with the appropriate values and calls the openDoor function if the user is allowed in the meeting.

Pre-conditions:

- The results from the Gateway should be received.
- The users' phone should still be on the Node.

Post-conditions:

- The Protection App should have received the results.
- The lights should reflect the colour of the results.
- The door should be unlocked if the lights turned green.

2.4.2.3 setColor

Priority: Important

Service Contract: This function receives an RGB colour value and sets the lights to turn that colour.

Pre-conditions:

- The lights should have power.

Post-conditions:

- The lights should be the desired colour.

2.4.2.4 openDoor

Priority: Nice-to-have

Service Contract: This function unlocks the door temporarily so that the user can enter the meeting room.

Pre-conditions:

- The door should be locked.
- The user must be in front of the door.

Post-conditions:

- The door should be locked again.
- The user must be on the other side of the door.

2.4.2.5 waiting

Priority: Important

Service Contract: Runs the lights through a sequence and delays the loop function a bit.

Pre-conditions:

- None

Post-conditions:

- None

2.4.2.6 setup

Priority: Critical

Service Contract: Initialises the necessary variables and objects and starts the loop function.

Pre-conditions:

- The Node should be plugged in.

Post-conditions:

- The necessary variables and objects should be created in memory.
- The loop function should have been called.

2.4.2.7 loop

Priority: Critical

Service Contract: Gets automatically called over and over like an infinite loop. Each time this function executes it calls the waiting function and then checks for input from the NFC board. When a valid identifier is received, it calls the sendMessage function with the results from the isAllowed function.

Pre-conditions:

- The setup function should have succesfuly completed.

Post-conditions:

- Either Nothing should have happened,
- or the user should have his access granted or denied.

2.5 Gateway

2.5.1 Scope

- The Gateway receives the device ID from the Node. It then verifies the persons access to the meeting and sends a response back to the Node.
- The Gateway synchronises with the server and makes a local copy of the meeting. The local copy minimises the delay and complexity.
- The Gateway updates the meeting on the server.

2.5.2 Functionality

2.5.2.1 sync

Priority: Critical

Service Contract: Synchronises the content on the server with the copy in the local cache.

Pre-conditions:

- The Gateway must be connected to WiFi.
- The Server must be running on the domain.
- There must be at least one meeting set up.

Post-conditions:

- The local cache copy of the meeting(s) should look exactly like the one on the server.

2.5.2.2 receive

Priority: Critical

Service Contract: Gets the device ID from the Node via the Serial connection.

Pre-conditions:

- The Node should be connected to the Gateway with a Serial connection.
- The Node should have scanned a phone and have sent the device ID.

Post-conditions:

- The device ID should be stored in a local variable.
- The verify function should be called.

2.5.2.3 verify

Priority: Critical

Service Contract: It calls the sync function if it's necessary and then checks whether the device ID is allowed into the meeting or not..

Pre-conditions:

- The receive function should have successfully completed.
- The sync function should work.

Post-conditions:

- The send function should be called with the results.

2.5.2.4 send

Priority: Critical

Service Contract: It returns the results via Serial connection to the Node and updates the information on the server if necessary.

Pre-conditions:

- The Gateway should be connected to WiFi.
- The Node should be connected to the Gateway via Serial connection.
- The verify function should have successfully completed.

Post-conditions:

- The Node should have received the results.
- The server should be up to date on who logged in or out.