Unsolvable Solutions
Client: Francois Mouton at the CSIR DSSR

Architectural Requirements

Eavesdrop

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 **2**
- 1.1 Project Background . . . 2
- 1.2 Project Vision . . . 2
- 1.3 Project Scope . . . 2

2 Access Channels **2**
- 2.1 Human Access Channels . . . 2
  - 2.1.1 Server Application . . . 2
- 2.2 System Access Channels . . . 2
  - 2.2.1 Server Application . . . 2

3 Integration channels **3**

4 Quality Requirements **3**
- 4.1 Scalability . . . 3
- 4.2 Performance . . . 3
- 4.3 Reliability and Availability . . . 3
- 4.4 Portability . . . 3

5 Architectural tactics or strategies **3**
- 5.1 Thread pooling: . . . 3

6 Architecture Constraints **3**

7 Technologies **4**
- 7.1 Java . . . 4
- 7.2 Android SDK . . . 4
- 7.3 TCP/IP . . . 4
- 7.4 User Datagram Protocol . . . 4
- 7.5 WebView . . . 4

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 have access to 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 malicious software(malware) on their smart phone, it could cause sensitive data to be easily leaked out.

1.2 Project Vision

The aim of the malware is to eavesdrop on a person via their own smart phone. This is done by live streaming the conversation from the infected smart phone to a remote server which plays back the stream and saves a local copy of the recording on the server.

1.3 Project Scope

The malware consist of a web server that connects with the mobile device via a web socket. A request is send from the server to the application on the mobile device to start recording. The mobile device will start to stream live to the server which in turn plays back the stream and creates a local recording.

2 Access Channels

2.1 Human Access Channels

2.1.1 Server Application

The malware application can be controlled using the server application. The server will communicate with the malware application with user input and receive the live audio stream.

2.2 System Access Channels

2.2.1 Server Application

The server application interacts with the malware application with the use of sockets. Commands are sent to the malware application through the connection between server and the malware.

3 Integration channels

- The malware is embedded in a application that is most likley used by the target. The user can control the malware application through the server application that runs on a remote computer.

4 Quality Requirements

4.1 Scalability

The server can handle multiple connections at a time of which a list is created. One device can then be selected from the list from which the live stream will be requested. The server can only handle one live stream recording at a time.

4.2 Performance

The malware application will have a minimal effect to the application it is embedded in such that it is almost invisible. The server will have a number of devices connected but only one device can stream at a time. I will have a minimal effect on the host machine. There is a limit on the recording time as the device will record until the user stops the recording.

4.3 Reliability and Availability

The malware application will be reliable because if the user closes the application the malware will run as a background task.

4.4 Portability

The malware server is easily portable as it can run on any computer running java.

5 Architectural tactics or strategies

5.1 Thread pooling:

It is used to improve scalability, to enable connections to a number of devices from which one device can be selected to create the live stream.

6 Architecture Constraints

- The Android malware application requires a minimum version of Android 4.0
- The malware server application requires a minimum of Java 7.

7 Technologies

7.1 Java

Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java was used because of its Audio package which provided the tools to manipulate the audio stream and to create the local recordings.

7.2 Android SDK

A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Abdroid SDK was used because a native android appication was developed.

7.3 TCP/IP

Transmission Control Protocol/Internet Protocol, TCP/IP is the suite of communications protocols used to connect hosts on the Internet. This technology was used for communication between the malware application and the server.

7.4 User Datagram Protocol

User Datagram Protocol(UDP) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet.
UDP was used because UDP packets are asynchronous unlike TCP where if a packet gets lost it will request the packet again. UDP is required because it will stream live audio and if a packet gets lost it will continue to stream the audio.

7.5 WebView

WebView is a browser bundled inside of a mobile application producing what is called a hybrid app. Using a webview allows mobile apps to be built using Web technologies (HTML, JavaScript, CSS, etc.) but still package it as a native app and put it in the app store.
This technology is used because as proof of concept it proves the malware can be embedded in a application in this instance it is hidden behind a WebView browser.