Stolen Object Tracker

Cycle 1 Report

By

Charles Baker

Denney Burkholder

Nathan Plotts

Leo Reyes

Steven Whaley

Submitted in partial fulfillment of the requirements for  
COMP 4710 Senior Design  
to the Department of Computer Science and Software Engineering,  
Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

March 4, 2014

Table of Contents

[2 Executive Summary (System Metaphor) 4](#_Toc381373422)

[3 Project Introduction 5](#_Toc381373423)

[3.1 Previous Development 5](#_Toc381373424)

[3.2 Intent This Cycle 6](#_Toc381373425)

# Executive Summary (System Metaphor)

By: Leo Reyes

When an object is stolen, it often never makes it back to its owner. Although there are other tracking devices currently on the market, our product provides more data about the thief than a regular tracker can, and also provides a user-friendly online interface for viewing device related updates and managing user accounts.

There are two different tracking device options. The first is a laptop application that runs in the background, hidden from the user. When the laptop is stolen, the application can gather information from its environment and use this information to locate and retrieve the laptop, this information includes their IP addresses as well as a record of their keystrokes.

Another tracking option available is a GPS tracking device. This device is small enough that is can be placed inside most commonly stolen objects and will activate when notified by the owner.

A user-friendly online system, with a simple account registration and login, is available to device owners. When a device is stolen, the owner can easily activate the tracker by logging into the online system. The owner of the device can view location updates on a map through the online system, along with data related to specific devices including logged keystrokes and IP addresses. Support is also provided for registering and managing multiple devices.

# Project Introduction

By: Denney Burkholder

Over the past four weeks, our team has been working on creating a recovery program to track objects that are stolen without alerting the thief that he or she is being monitored.

We are designing for laptop computers and an object location device.  The goal for our project has been to create a product that will allow our customers to view their locations through our website, along with other information that may prove useful to recovering the device.  The need for a product such as this is undeniable when looking at the recovery rates for expensive devices (such as laptops, mobile phones, and tablets) once they have been lost/stolen.  Anyone who owns a desirable device can fall victim to thievery.  As such, they are our intended clientele.

By logging on to our user-friendly web application, the customer will be able to register their devices and access different information depending on the type of device.  For laptop computers, we have created a key logger that will be remotely activated by our server once the device has been reported missing on our website.  It runs in the background of the laptop, so the thief will not be aware that information is being gathered.  This file will be sent to our server whenever the laptop can get an Internet connection through available Wi-Fi networks.  The files can be viewed and downloaded from our web application. Our object location device is a self-aware tracking device.  It has GPS capabilities and an accelerometer allowing it to transfer its location along with the speed with which it is traversing to our server for viewing on our web application.  It communicates through cell towers, so a Wi-Fi connection is not necessary.

 Implementing our project involves many different components.  We have designed and coded a central server, database, applications for gathering data on laptops and web pages for communicating between the stolen devices and customers, and storing relevant information for recovery.

## Previous Development

By: Charles Baker

Previous development was performed in the architectural spike phase. Development was focused on the server and Windows service. Goals for the architectural spike were to get the most important features to a point where it was evident we could continue development without any major problems that may cause us to miss our final deadline.

Goals for the server were to get most planned functionality working in a basic state. The server was designed in 2 primary parts, the application server and web server. Goals for the web server included: embedding Google Maps into a page, setting markers on the map, and connecting to the application server. Goals for the application server included: communicating with the Windows service, communicating with the database, communicating with the web server, and moving data between different connections on request. The goal to connect the web and application server was not met because of difficulties with GlassFish. Furthermore, we were not able to work on sending requests do to time constraints and because most involve the web server. Otherwise, we made satisfactory progress on the remaining goals. We created the html pages and were able to embed Google maps into it. We were also able to add markers to the map by sending coordinates through a web socket. The application server was able to communicate to the Windows service by a TCP connection. We developed a messaging protocol that both server and client (service) adhere to when communicating. The server was also successfully communicating to the database by the end of the architectural spike. We were able to read and write to a database containing all values we predicted we would need in the future.

On the Windows service, by the end of the architectural spike we wanted to have it start automatically, communicate with the server automatically, identify itself, and log keystrokes. By default, when the service is installed it launches on startup. The service was able to identify itself by using a MAC address of a network adapter. This method mostly worked, but a virtual adapter appearing could cause problems in selecting the same MAC address each time. Connecting to the server was achieved by creating a thread to handle the connection. If connection was ever lost, it would simply try to reconnect. While connected, the service could respond to different commands sent by the server. Commands implemented where: start key logger, stop key logger, return key logs, and send an IP trace. Key logging was achieved, but was in a very basic state at the end of the architectural spike. It was able to record the key being pressed, but did not consider keys that were being held down. This means that it did not record the case of letters or other effects of key combinations being pressed.

## Intent This Cycle

By: Charles Baker

We decided it would be beneficial to port the existing server to the Google Go language. This allowed us to have a single server, instead of both an application server and web server. This is possible because Go allows application code that contains HTTP handlers. It also provided better support for multithreaded applications, which is important for our server that must manage connections to many devices and web connections. We believed that In the long run, the change to Go would speed up the development of the server by requiring a fewer number of components that must interact.

Another one of our primary goals this cycle was to incorporate the Geogram One into the system. The first step was to send messages between the server and Geogram One. We decided the communication would be done through an Android phone. An Android application would be responsible for connecting to the server via TCP and interfacing between TCP and SMS communication. The server must be able to send the appropriate commands and interpret received messages.

Work on the Windows service was also planned for this cycle. We planned to have the service check in to the server in set interval. If the server notified the service that the laptop was stolen, it would constantly try to reconnect if the connection is ever lost.