**Stolen Object Tracker: Continued Development**

**Server (whole):**

The server is written in the Go programming language. Many of our group members used liteide to work on the server. You can also use a text editor (like Sublime Text) and compile via command line. When compiling from command line you’ll have to set the GOPATH variable to the folder that contains your src, bin, and lib folders. We had issues when compiling from command line and having the project build to the bin folder. We solved this by going to the /src/server directory and running “go build server” which builds the executable in the /src/server directory. This is helpful because the css, images, js, media, and templates folders must be in the same directory as the executable when it is ran.

There are various improvements that could be made to the server at this time.

Not all requests that are handled respond to the sender of the request. Adding responses to all requests would improve the reliability of the system. To see an example of how the request/response system works, look in the webServer.go at the login handler. The webServer sends a request to the database and, depending on the response, will log the user in or tell the user the entered credentials are invalid.

The server could use a logging system. The easiest place to start would be logging all requests that are sent and the responses to them. All requests pass through the central server, so that would be a good place to capture them. Additionally, some events between the deviceHub and laptopHub, and between the deviceHub and gpsHub, could be worth logging.

**Database/Database Controller:**

Keylog data or possibly the whole database should be encrypted.

**Web Server & Website:**

The web server needs to communicate with the website to get the Geogram’s PIN upon registration. At the time being, the service uses the Geogram’s default PIN, 1234. Security could be improved by allowing users to specify their own PIN.

Traffic between the server and website should be encrypted.

**DeviceHub:**

Communications between laptops and the server should be encrypted.

**Windows Service:**

The Windows service was developed in visual studio. Opening the solution in Visual Studio to continue development.

To debug the service or key-logger, run them as you normally would. Use the “Attach to Process…” option on Visual Studio.

The Windows service is left in a very good state. The only functionality that we were not able to implement was connecting to the unprotected Wi-Fi automatically to increase the likelihood of finding a stolen laptop. Also, traffic between the service and server is not encrypted. It would be beneficial to encrypt this traffic in the event that the user reports the laptop stolen and forgets to deactivate it when recovered. If the key-logger is still running, their data will not be sent over the Internet unencrypted.

The key-logger is also in a good state. One thing that could be done is to handle certain key combinations that are used as shortcuts, such as ctrl + s to save. Otherwise, most keys and key combos are captured. Additionally, using keys that don’t produce characters, such as arrow keys, can make the logs difficult to comprehend. A way to filter out those keys could be beneficial, although, that could also be done server-side.

An improvement that could be made to the key-logger, is to display the name of the window (or what application it is) in which the user is typing.

**SMS Gateway:**

The SMS Gateway app was developed in Eclipse using the ADT(Android Development Tools) plugin. The easiest way to continue development would be to download the ADT bundle which has eclipse in it with the plugin preinstalled.

The SMS Gateway app could be continued by checking that the text is successfully sent and reporting it to the server. Currently the server is unaware of problems that occur on the app.