RoboTractor: A Secure Signaling and Control System for Remote Management of Agricultural Vehicles using XMPP and REST

Jeremy Wright
Arizona State University
jlwrigh1@asu.edu

Arun Balaji Buduru
Arizona State University
abuduru@asu.edu

David Lucero
Arizona State University
dwlucero@asu.edu

***Abstract*—The need for the automation of vehicles is increasing due to improved efficiency and reduced operational costs. The scope of this project is to build a secure signaling and control system with mechanisms to drastically reduce if not eliminate the secure signal and control systems from being are compromised, making automated vehicles vulnerable to be stolen, damaged, or otherwise harm the environment. The main tasks in this project are to build a front-end interface for users to give instructions built upon a Representational State Transfer (REST) architecture, set up Extensible Messaging and Presence Protocol (XMPP) server and client with required authentication and data integrity check mechanisms, and build a simulated automated vehicle, capable of interfacing with the XMPP components and front-end.**

***Index Terms*—Secure signaling and control, remote management, agricultural vehicles automation, self-driven vehicles**

I. Introduction

The most important requirement in developing a secure signal and control system is ensuring authentication of vehicles and data integrity, here the path plan and location feedback transmission needs to be secured and protected from eavesdroppers, preventing attackers from compromising and/or stealing these vehicles. In this project we use Django to handle the front end interface for the users. We use python libraries to interface between user and path generator server where we use XMPP and REST protocols, and XMPP clients to control the agricultural vehicles. We expect to have a comprehensive tool that translates the abstract user inputs into executable actions for agricultural vehicles in a secure manner.

II. System Models

A. System Model

RoboTractor will leverage the Django Web framework [1] , to realize the required interfaces. Figure 1 describes the connection of these components. The combination of XMPP and REST in RoboTractor is a demonstration of how to extend the existing HTTP development environment i.e. ”the web of things” into a stateful protocol. HTTP is by design stateless. XMPP on the other hand is a stateful streaming connection between two clients. In this case the Tractor and a command server. To achieve this mesh we will leverage the Bidirectional-streams Over Synchronous HTTP standard protocol [2]. BOSH provides a standard mechanism to operate the streaming XMPP protocol efficiently over an HTTP connection. This is essential for a scalable webservice.

B. Software

1) XMPP Server: RoboTractor uses the ejabberd [3] XMPP server as it provides an existing Python interface, to integrate with the rest of our Python based ecosystem.

2) REST Interface: TastyPie provides REST [4] by extending the existing Django Models.

3) BOSH Interchange: Punjab is an Django plugin implementation of BOSH [5]. In addition to BOSH, this library combines the ejabberd Users with Django Users to provide a single authentication and authorization framework. While this demo project will have a single user type, this combination is critical to maintain proper use management and least-privilege authorization.

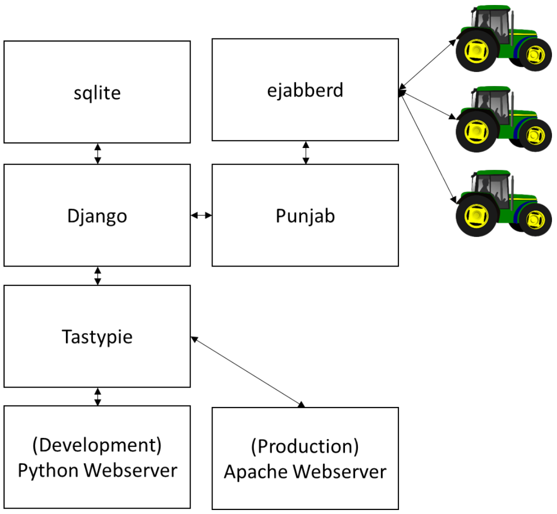

Fig. 1. Software Component Block Diagram

C. Security Model

RoboTractor deals with two primary security classes: One involving the vehicles and other assets, and one involving the system components and their communication channels. Because the project is web-based and uses PKI, numerous attack models have been created that this project will deal with. Within the class of communication channels, attacks may be present against the authentication systems of both users and assets. Robotractor will employ advanced two-factor authentication. Usage of TLS and other best common practices will help to prevent against replay attacks on the system, and also for establishing a secure session between assets and control. A dual-encryption scheme has been devised to send encrypted control signals within an encrypted XMPP stanza to ensure high information assurance. Because this project will utilize PKI, key protection is important. To this end the RoboTractor project plans to secure keys on assets with a Trusted Platform Module or relevant simulated facsimile. By making both key-pairs private and secure, RoboTractor provides advanced communication security. On the other end, the server components must be secured as well using best common practices in server defense, firewall, patches, updates, etc. Additionally, the RoboTractor system will implement an advanced GPS location reporting system to provide physical security in case of asset theft. By defending against these attack models, the RoboTractor project presents the realization of an advanced system for signaling and control with high security.

III. Project Description

Implementation is divided into 2 primary phases (Marked as milestones on the attached Gantt chart). The first phase is an integration phase connecting the various libraries and off the shelf components. It is critical that this integration does not weaken, but rather strengthens the security properties provided by each independent component.

The second phase is the extension of these components to provide command and control of agricultural vehicles. For schedule mitigation, David and Arun are scheduled to begin this phase before the framework is assembled. Once the framework is in place, David and Arun's interface designs may be merged with the overall system.

A. Task 1 : Development Environment

The development environment consists of configuring all the tools needed to portably work with the software package. This will involve installing project dependencies, and deployment scripts to allow all members of the team to work effectively together. The complete environment will be stored in git.

B. Task 2: XMPP Implementation

XMPP Implementation is a configuration task to setup the ejabberd server and connect it into the HTTP Framework. Once this is in place the XMPP Clients may start working.

C. Task 3: Demo Tractor

The Demo tractor is the complementary component to the Frontend UI. This is the virtual machine, a piece of software which simulates a real tractor, or agricultural vehicle.

D. Task 4: Frontend UI

The ”single-page” web application is the modern design methodology to web apps today. Leveraging this design architecture the Frontend will query the REST API to draw the position of all tractors within a Google Map context. This task has 2 high level sub tasks:

1) Path generation

2) Map rendering

Path generation is the primary deliverable of this project. The user shall be able to input a path for a given tractor to drive. The Tractor will then receive this path over the REST to XMPP gateway. The UI may then periodically query the position of the tractor and update it on the Google Map.

E. Task 5: Server Backend

The Server Backend is the critical component which plumbs all the components together. Extensive knowledge of Django will make this task easier. As it will require linking multiple components together in a orchestrated fashion.

F. Task 6: REST Interface

The REST interface will server the primary means of interacting with the site. The Web interface will exist as a ”single-page” app who leverages AJAX principles over this REST API.

G. Project Task Allocation

From the tasks outlined in section III, the breakdown of work will be as follows: Jeremy Wright will work on configuring the development environments and the REST Interface. He will operate as the project lead due to his experience in Python-based Web Applications. Arun Balaji Buduru will work on the Server Backend and Frontend UI components. David Lucero will work primarily on the Demo Tractor agents and also XMPP implementation. Each group member will have approximately 30% workload of the project with the remaining 10% shared due to the interconnected nature of all components. A detailed task breakdown can be seen in the attached Timeline.

H. Deliverables

Midterm Deliverables:

1) Functioning REST Interface
2) Front-end Test Interface
3) API documentation for using the REST interface as an external service.
4) API documentation for using the XMPP interface to act as a vehicle.
5) Finalized Design of Vehicle simulator
6) Interim Progress Report
7) Updated Project Schedule

Final Deliverables:

1) Finalized back-end configuration
2) Administration panel for configuring vehicles.
3) Finalized front-end interface
4) Vehicle Simulator with working component interfaces
5) Final Project Report

I. Project Timeline

The attached project timeline (generated from Microsoft Project) describes the overall tasks of the project.

IV. Risk Management of the project

Several potential issues have been identified that may pose risk to successful completion of this project. These risks have been identified in Table I. Along with the description, ratings have been assigned to each risk identified, and potential mitigation strategies are described.

V. Conclusion

In this proposal we intend to build a remote control system leveraging existing internet technologies XMPP and REST. Future work based on this approach may include asset management for corporate farm management.

References

[1] (). Django, GitHub, [Online]. Available: https://github.com/django (visited on 02/06/2014).

[2] I. Paterson, D. Smith, P. Saint-Andre, and J. Moffitt. (Jul. 2, 2010). Bidirectional-streams over synchronous HTTP (BOSH), [Online]. Available: http://xmpp.org/extensions/xep-0124.html (visited on 02/06/2014).

[3] (). Ejabberd community site | the erlang Jabber/XMPP daemon, [Online]. Available: http://www.ejabberd.im/ (visited on 02/06/2014).

[4] (). Toastdriven/django-tastypie, GitHub, [Online]. Available: https://github.com/toastdriven/django-tastypie (visited on 02/06/2014).

[5] (). Twonds/punjab, GitHub, [Online]. Available: https://github.com/twonds/punjab (visited on 02/06/2014).

TABLE I
RISK MANAGEMENT

Risk Description	Risk of Failure	Consequence of Failure	Mitigation Strategy
Connections between components must be secure	Low	Product would still function, but would lack security information assurance will suffer	Plan to include security on component connections ahead of time. Follow best practices (SSL/TLS)
Evaluation of Project depends on proper vehicle agents	Medium	System is unable to be tested if vehicle agents are inoperable or incorrect	Vehicle agents will be designed first to ensure compatibility with all other components. Thorough review and testing of component to ensure proper functionality
Improper configuration of components	Medium	Components do not communicate properly with each other. System functionality is void	Thorough reviews and testing of components to ensure proper functionality and communication
Incorrect PKI implementation and/or configuration	Medium	Puts all identified security domains at risk. PKI used becomes useless. System becomes vulnerable to outside attacks i.e. MITM	Follow best practices and standards. No re-invention of the wheel
Service Uptime (web reliably)	High	Product is unusable if network connections are not operable	Redundancy. Cloud hosting to alleviate hardware reliance