**Goals**

- Gain understanding of TR-069, scope of the project
- Identify open source project which most closely implements standard
- Reverse engineer to assure compliance with TR-069
- Document results
- Update code to remain compliant
- Demonstrate functionality with simulated clients
- Demonstrate functionality with Physical Optical Network Terminal
- Develop virtual machine running the server

**Overview**

**Work Breakdown Structure**

- **Amadu**
  - Compile LibreACS source code
  - ACS-Client Interfacing
  - Client setup and operation
- **Mason**
  - Database Analysis
  - UI Update
- **Donnell**
  - Database Analysis
  - UI Update
- **Robert**
  - Reverse engineering, documentation
  - Scripting Interface
  - Control flow
  - Virtual Machine maintenance
- **Alex**
  - Reverse engineering, documentation
  - Messaging
  - Control flow

**Background and General Information**

The ACS communication is defined by a number of protocols:

- **SOAP** – Remote Procedure Call
- **XML** – Data transmission
- **HTTP** – Packet communication

The ACS server’s back-end consists of a SQL database, which interfaces with our software through XML and Java EE libraries.

**Results**

**Backend**

- **MySQL**
- **TR-069 Protocol**
- **SOAP**
- **XML**
- **HTTP**

**GUI**

- **CPE NULL request**
- **JSF**

**Conclusion**

We studied several existing open-source software packages and selected one for our project. We hosted the selected packages on Virtual Machines (VMs) running the Ubuntu 14.04 Linux distribution, documented it, finalized the code to be fully compliant with standard, and tested it with a Python client, and a Java based client.

**Impact**

The ACS server and CWPE client will be used by iPhotonix to connect millions of network devices across the country and beyond. Due to our work there is a probability that you will use their device in the near future.