Problem Statement:

While some newer SCADA systems are designed with security in mind, there is a lot of legacy hardware and software out there that were designed and implemented before security was a concern. Due to the cost and long life of these systems there are many still in use today that are 10-20 years old, that were designed 20-30 years ago. Replacing them wholesale is far too expensive, and replacing them piecemeal doesn't solve our security concerns.

This leaves a lot of vulnerable SCADA components that are not sufficiently protected in the wild. Crafting software that can demonstrate these vulnerabilities will hopefully lead to better security as vulnerable systems are shown to have problems.

Goals:

Develop a piece of software that takes advantage of a security flaw to change the operating parameters of a piece of SCADA hardware.

Objectives:

Analyze existing SCADA code for flaws or assumptions that aren’t safe in an adversarial environment.

Determine which of those flaws or assumptions would provide the best to attack or defend.

Write software that will intercept or inject traffic between SCADA systems to change the operating parameters and hide those changes to the monitoring system.

Test this software in a virtual environment to demonstrate its functionality.

Scope:

Our project is operating under the assumption that a flaw or security oversight exists such that we can run the attack code in a MITM configuration between the monitoring systems and the reporting field elements. The method by which this software is installed or covertly communicated with is beyond the scope of the project.

Tentative timeline:

28th September	- acquire SCADA software and determine type of test environment to simulate.
5th October	- analyze normal communication protocols for flaws and security oversights.
12th October	- design 2 or 3 theoretical attack models that could work.
19th October	- Milestone presentation 2 – design and implement a packet capture / packet spoofing module for SCADA communications.
26th October	- design and implement system to record and playback normal traffic
2nd November	- design and implement system that will change the operation state of a end unit.
9th November	- Milestone presentation 3 – automate system to hide changes and read from configuration files.
16th November	- Finalize research and development
21st November	- prototype and/or outcome demonstration, draft project report
30th November	- Project report submission and project presentation

Serial Vs TCP/IP

Serial connections are in a series.
Serial packet is broken up into 3 main parts:

1) Slave address

2) Modbus application Programing Data Unit: (a function call, and possibly some parameters)
3) Error Checking:

The response packet is largely the same:

1) Slave address
2) Modbus application PDU: (function call/error code, and data or error information)
3) Error Checking

TCP/IP
Master(s) and Slaves are put on the same network and communicate over ethernet.
Packet Breaks down to:

1) Ethernet, IP, TCP frames
2) Modbus protocol
    a) Modbus application protocol
        i) Transaction ID (how you pair requests with responses)
        ii) Protocol (determines the protocol, in this case Modbus)
        iii) Length (how long is the rest of the packet)
        iv) Unit ID (the ID of the slave)
    b) Modbus application PDU

Because neither of these have any sort of accommodation for authentication or encryption, any packet can be spoofed or intercepted. This leads to all kinds of exploits if there is a bad actor on the network.