Implementing OpenPLCs into a Cyber Defense Competition

DESIGN DOCUMENT

Team 16

Dr. Doug Jacobson
Dr. Julie Rursch

Nick Springer - Security Engineer
Matthew McGill - Project Manager
Val Chapman - Software Testing Engineer
Josh Przybyszewski - Software Engineer
Joseph Young - Software Engineer
Liam Briggs - Hardware Engineer
Brennan Ferguson - Hardware Engineer

sdmay18-16@iastate.edu
http://sdmay18-16.sd.ece.iastate.edu

Revised: 12/4/2017

Table of Contents

List of figures/tables/symbols/definitions

Introduction

1.1 Acknowledgement

We would like to acknowledge the assistance provided to us by Doug Jacobson and Julie Rursch throughout the duration of our project. We are extremely grateful for the time, help and equipment provided to our entire team.

1.2 Problem and Project Statement

At core of our project, is a desire to create a system to simulate a Cyber Physical hardware with the OpenPLC platform. The CDC has provided a great opportunity for students to learn concepts related to Operating Systems, security and even SCADA systems. With the OpenPLC platform, we would like to build a system that will allow the participants to also learn about Cyber Physical systems. With the increasing use of computing techniques in physical systems, it is important to teach security concepts related to these systems.

The project is an effort to create systems for the Cyber Defense Competition that will simulate Cyber Physical concepts. Specifically, we are going to simulate a factory floor that the Cyber Defense teams will be responsible for managing. Furthermore, the project will utilize the the OpenPLC platform to interact with the physical systems that we will simulate.

In the end, we would like to have a project that can be integrated into the Cyber Defense competition. Specifically, we would like to have servers setup that can be copied and integrated into a CDC with relative ease. The servers that we setup will have a majority of the system setup, but will include security flaws and other issues the Blue Teams (see section 1.4 for definition) will need to resolve. These servers will host the Factory.IO software that virtualizes a factory environment and the web servers that will be used to interact with the factory.

1.3 Operational Environment

We will be virtualizing our system, so our project will exist on the same server that hosts the ISEAGE environment. Therefore, we will not need to worry about physical conditions because the server is already hosted in a secure and appropriate environment.

1.4 Intended Users and uses

Blue Team

Students participating in the Cyber Defense Competitions are responsible for completing set up and securing the systems provided to them via the competition scenario. Blue teams make up the largest demographic of active participants. Each blue team ranges in size from 5-8 individuals. From the perspective of the proposed project, each blue team will receive an individual network of servers revolving around the Factory I/O platform. Their primary job is to become familiar with interfacing with the PLCs within Factory I/O, as well as get them fully functioning and operational. Security will become a priority, as the Red Team will be on the prowl during the competition, looking to take down individual systems and disrupt the manufacturing process.

White Team

The White Team is responsible for running the entire competition the day of, generating the anomalies, maintaining systems, etc. This committee of students distribute the network of servers to each Blue team, overseeing official competition rules, and keep all of the working parts running together smoothly. As our project team gets closer to integrating the virtualized PLC environment in with an actual CDC, we will be working closely with the white team designated for that competition to sure everything is implemented properly.

Red Team

The Red Team is responsible for launching attacks against each of the blue teams throughout the day of the competition, striving to gain access to vulnerable systems, steal flags, and wreck havoc - attempting to make the lives of the blue teams miserable. The Read Team is comprised of security "experts" - including past students, industry professionals, security analysts, penetration testers, etc. By introducing the OpenPLC project into future CDCs, this will have a great impact on the way the Red Team interfaces with each of the Blue Teams. Instead of trying to exploit vulnerable web servers running older versions of outdated software, they will be targeting actual physical systems (through virtualized means). This adds additional layers of difficulty/complexity, as the end result will not be to simply shut down these physical systems or "gain root access", but figure out other ways to cause real-life, practical damage.

Company Sponsorship

Companies have the unique opportunity to become sponsors of current CDCs. Many of them contribute financially, as well as provide employees to compete with the Red Team. They also are on-site the day of the event, recruiting the next generation of cyber security talent, collecting résumés, scheduling interviews, meeting with students, etc. However, very few companies ever get the ability to help design competition scenarios. Their involvement with the competition itself is limited to day-of activities. With designing Factory I/O, we will now allow employers to design different scenarios right alongside us, modeling various cyber-physical systems that occur in their own manufacturing facilities. Employer interest will hopefully increase dramatically with the progressive move towards utilizing the OpenPLC project, and modeling real-life physical situations.

1.5 Assumptions and Limitations

Assumptions

There are several assumptions to be made regarding our proposed design, with these assumptions primarily revolving around the fact that our project will be tightly integrated with the Cyber Defense Competitions (CDCs) held at Iowa State. There will be a maximum of four teams

interacting with our project at any one given time (white team, green team, blue team(s), and red team). Each team will have a significant number of users, so our project will be modularized to accommodate the fluctuation each semester. Factory I/O will be the platform our project is based upon. This industry leading software platform allows us to virtualize complex environments and emulate real-life situations. Security will be a major focus in this project, with a primary emphasis based on securing these virtualized systems and protecting them from outside attack (from the red team).

Limitations

The system must live inside of a containerized network environment called ISEAGE (all CDCs occur within ISEAGE). The final product must be built on top of the OpenPLC project. The OpenPLC project is the first fully functional standardized open source PLC. It is a standard industrial controller built upon open-source hardware and real time responses. This platform has been selected for its standardized nature, and for being open source for continued long-term development. While this was not a strict limitation, the ability for this system to exist in a virtualized form was strongly encouraged by our faculty members, for scalability and cost reasons. Since we can assume over 30 blue teams would need their own, individual system, this made perfect sense.

1.6 Expected End Product and Deliverables

The end product for this project will consist of the following:

- Network of servers. The exact specifications of these servers and their intended functions can be found in section 2.1 of this document, where a breakdown of every service is completed.
- Formalized documentation for our findings on the OpenPLC project, exact step-by-step procedures for building particular systems in Factory I/O, detailed explanation of the intentional vulnerabilities/loopholes we've designed into each system, etc. The list goes on and on, but documentation will surely play a vital role in this project.
- Other ideas of PLC integrations with the CDCs, and what role the OpenPLC project will play in the process of adding these.
- Modularized network environment that is easily scalable and designed for use of many teams, with many inputs, with many external variables.

These products will be deliverable by next spring of 2018, around the April timeframe. Because the CDCs occur earlier in the semester, the first opportunity and CDC to take advantage of the integration of PLCs will be fall of 2018, in which our system will be up and running, fully operational and documented (for future senior design project teams to continue work where we left off).

Specifications and Analysis

2.1 Proposed Design

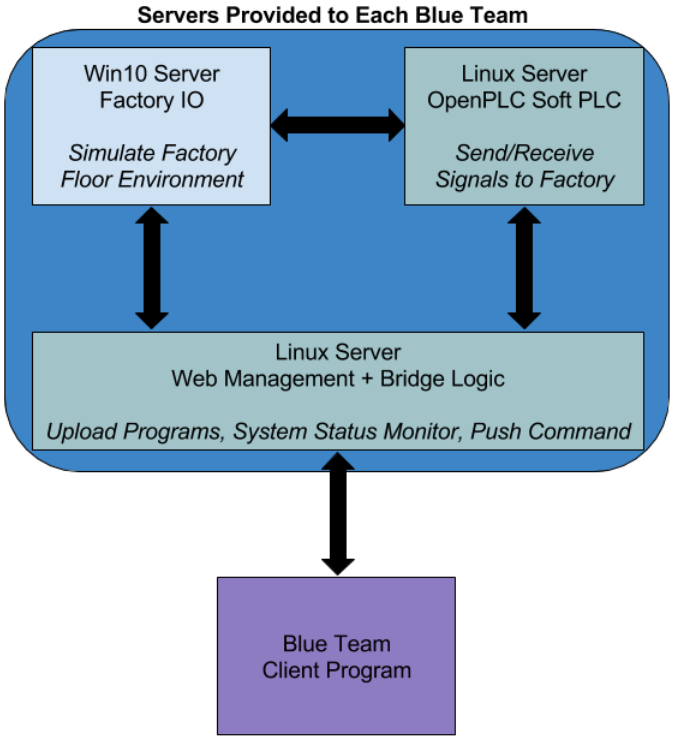

Figure 1

In designing our implementation of the OpenPLC project into the Cyber Defense Competition environment, we have made a shift from our previous hardware-centric focus, which simulated PLCs using Raspberry Pi 3 and Arduino units to manipulate real physical systems, to our current software driven model. This software-centric model utilizes one or more Linux servers to host a virtualized OpenPLC (soft PLC) and Web Management Server allowing the user to interact with the PLC by monitoring its outputs, sending it commands, or uploading new programs for execution.

Our design shall provide a base platform that includes the above components (soft PLC + Web Management server + Win10 server hosting Factory I/O) and is extendable for producing a variety of potential CDC scenarios with cyber physical systems. This may be accomplished by changing the virtualized hardware used within the Factory I/O software, or by replacing this Factory I/O module with different simulated or physical hardware entirely.

In addition, our design shall provide a functional simulation of PLC hardware and the systems they control via common protocols used by real-world PLC's, such as Modbus; furthermore, they shall be fed instructions using traditional methods such as Ladder Diagrams.

In researching solutions for simulating the systems controlled/monitored by the PLC, we discovered Factory I/O, which is a software solution that models many pieces of machinery to simulate the environment of a factory floor. This will provide "eye candy" for Blue Team participants; not only does this simulated environment allow participants to visually check that their team's systems are functioning properly, but also serves as a fun and engaging prop for relating their work to real-world applications. Factory I/O's services shall be available via Windows 10 virtual machines. To provide steady rendering of simulated factory floor equipment during a CDC (normal use case), ISU's Hypervisor shall be upgraded to include several modern graphics processing units.

Virtualizing OpenPLC on Linux, rather than exclusively using the Raspberry Pi + Arduino format, has the advantage of allowing our team to accomplish our goals while simultaneously reducing the hardware requirements for each CDC team; if we do not need to provide each team with a Raspberry Pi, our scenarios become scalable for varying competition sizes. For a stable installation, a current up-to-date distribution of Linux shall be used. However, vulnerabilities may be introduced for the Blue Teams to repair by intentionally providing them with outdated Linux distributions or software versions.

An additional Linux system may be used to serve the Web Management portal; when creating the minimal number of virtual machines is desirable, these services may be hosted alongside the soft PLC. Similar to the above, stable configurations should utilize up-to-date software for serving the website and providing intermediary logic between Blue Team clients and the OpenPLC instance. However, it may be useful to provide systems with outdated software to introduce difficulty into the CDC scenario for participants. The web client and any bridge logic should be modularly programmed to allow adaptation for future scenarios.

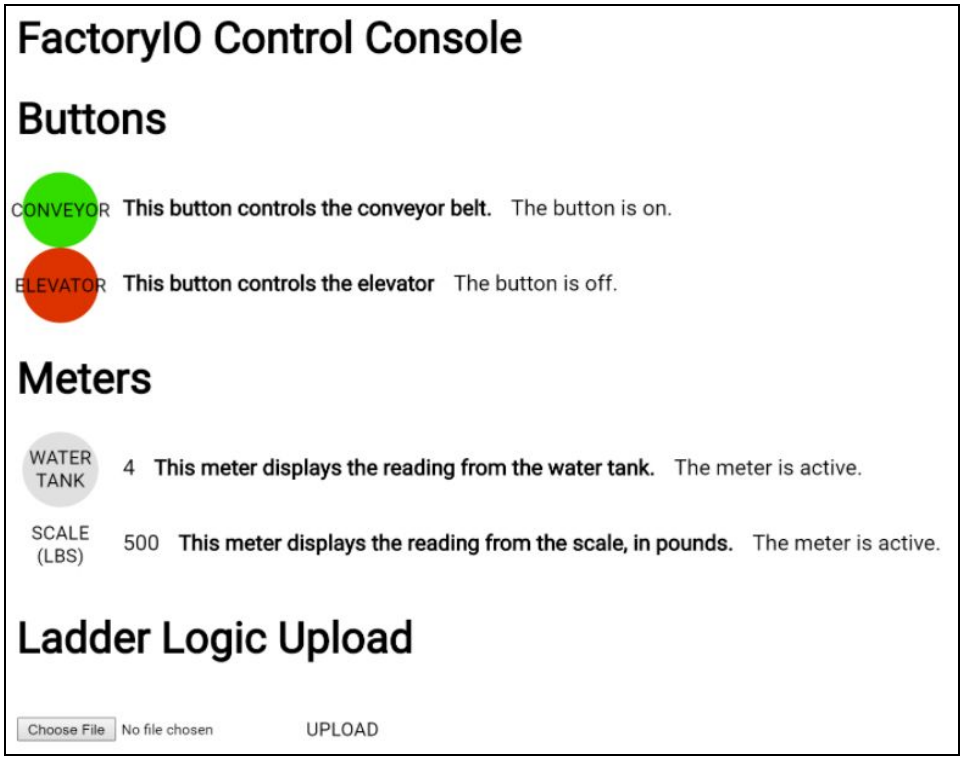

Figure 2

The web client, or web management portal, shall serve as an intermediary software for interfacing with the PLCs located within the Factory I/O software. Figure 2 is a mockup user interface that displays pertinent information to administrative users. This user interface shall display real time data and statistics of devices operating on the factory floor, as well as provide means for uploading new ladder logic files, allowing full customization and expandability. The user will also be presented with various controls, thus granting them the ability to turn PLC devices on and off (binary switches), observe trends in data, send various control signals, etc. The web management portal, as seen in Figure 1, interfaces with both the OpenPLC server (for the management of ladder logic files) and the Factory I/O instance, which houses the PLCs that our portal interfaces with. This will allow blue team users within the CDC to more easily manage their factory environments, as well as provide a potential attack vector for the red team to exploit vulnerabilities.

2.2 Design Analysis

In the early stages of this project, all the plausible ideas of the team were based on an implementation of a Raspberry Pi or Arduino computer. While most of the ideas met the basic project qualifications, they were not as complex or interesting as the team wanted. The initial idea was to use the OpenPLC software on a Raspberry Pi to power a model train and utilities on the track. After some research, the team decided against the train idea because of the limitations pricing and availability of model trains. Finding models that could be altered to be controlled by a PLC was not that difficult, as the hobbyists community is very active. Finding affordable versions of these proved to be very difficult. After abandoning this idea, we avoided requiring specific hardware that we wouldn't be able to create ourselves.

Keeping with a transport theme, the team discussed the idea of a traffic simulator. The PLC would control either self driving cars, stop lights, or other automated utilities. To represent the vehicles, programmable LED light strips would have been used. After much discussion, the team decided to move away from this idea as well. The end model would require a lot of back end code to work but the contestants in the CDC would have very little actual interaction with the model. It seemed as though the interaction for the users would be trivial and the project would be merely 'eye-candy.'

After discussion with our mentors with our concerns that the project could easily become purely cosmetic, the team decided to look for more physically dependant and interactive ideas. We came across some interesting elevator-like implementations based off weight and color of real world objects to take inspiration from. The team discussed the difficulties and freedom of building a custom mechanism like this and in the end decided against it because of the difficulty of scaling. Creating the individual elevators would be time consuming and unrewarding. If the competition had even one more team than elevators, they would not be able to be implemented. The team wanted the customizability of building but the scaling of something easily reproducible.

We found these qualities in a software deliverable. The team believes to have found the tools needed to accomplish them as well as create an interesting and effective representation delivery in Factory I/O. Using multiple systems to handle replicated interactions, all virtualized, creates enough interaction and complication to keep the blue team working for finding vulnerabilities as well as interesting and potentially challenging work within the software itself. The team has also discussed the many ways we could implement short cuts for red team to interact with the software for an even more challenging experience for blue team. We have had success in

implementing different components within the software and using other computers on the network to interact with the system and software. As we continue to test the possible interactions between the OpenPLC system and Factory I/O, we will discover the most optimal ways to take advantage of this software and find creative ways to implement the OpenPLC's and challenge the future blue team.

3 Testing and Implementation

3.1 Interface Specifications

Software Interfacing:

Our front end interface will have two uses. One aspect will be that the blue team will be able to use it to set up security, and have a UI to see the factory. The other side of user interface is to give us the development team a solid way to test that our Ladder Logic files are being uploaded to our SoftPLC correctly, and that signals are being sent to and from the PLC correctly. Using our front end UI, we will be able to test, the securities of our product and lack there of, the signals being sent to the PLC, and the Factory I/O Connection and display. These three portions of our project are some of the most vital portions of this project, and this interface will allow for connection and testing of all of them.

3.2 Hardware and software

Software Used In Testing:

Selenium (http://www.seleniumhq.org/):

Due to us having a front end that many users will be using and securing, we will want to have some automated tests, to make sure that the frontend is setup correctly. Selenium is an open source web interface testing software. We will be able to write scripts that check to see that any changes we make, do not change any of the required components of our front end.

Cucumber (https://cucumber.io/):

We will use Cucumber to make automated test cases that are in plain English to allow for other users to come along some time and use or fork our code, and make their own projects. We will use cucumber so when others are changing our UI, and web interfaces they will learn if they did break something, and will be able to narrow their errors down and hopefully fix them.

3.3 Process

We have used two methods to find our current solution we are implementing into a CDC. The first method we used to find our current implementation is researching into possible solutions that contain OpenPLC. The second method we have used to make our progress is testing and the viability of Factory I/O into our OpenPLC server connection.

A large part of project has been devising potential solutions and seeing if they are practical. We had four possible solutions, a train, light boards, elevator, and Factory I/O all controlled by our OpenPLC. We found that the trains were too expensive to implement. The light board, we

researched into the connection, and found that the complexity of programming the lights, and the control boards controlling them were too expensive, and very difficult to create multiple scenarios easily. The elevator was also rejected for the reason of difficulty to implement diverse scenarios. This testing and research led us to Factory I/O.

To test the Factory I/O we set up an OpenPLC server on a linux machine and then set up Factory I/O on a windows machine. For testing Factory we created a few scenes that are conveyor belts, moving boxes, to show that Factory works as we would like. We then set up a connection between OpenPLC and Factory and made sure we can receive connection calls between both systems.

Another aspect of testing that needs to be considered it testing the vulnerabilities on the services to ensure they are implemented properly. This testing can be achieved through the use of scripting languages like Python or Bash. Python could be used to verify that the vulnerabilities are correct from an external standpoint. For example, if we have SQL Injection on a website, the python could automatically do a check against the server with test input that proves the site is vulnerable. Bash could be used from an internal perspective to verify things like open ports, users, and malicious programs. Both of these approaches would be automated and useful when redistributing our services to ensure everything is still functioning as expected.

3.4 Results

Factory I/O can be used to display that a factory floor has been hacked, in this case, a scene of a failing factory:

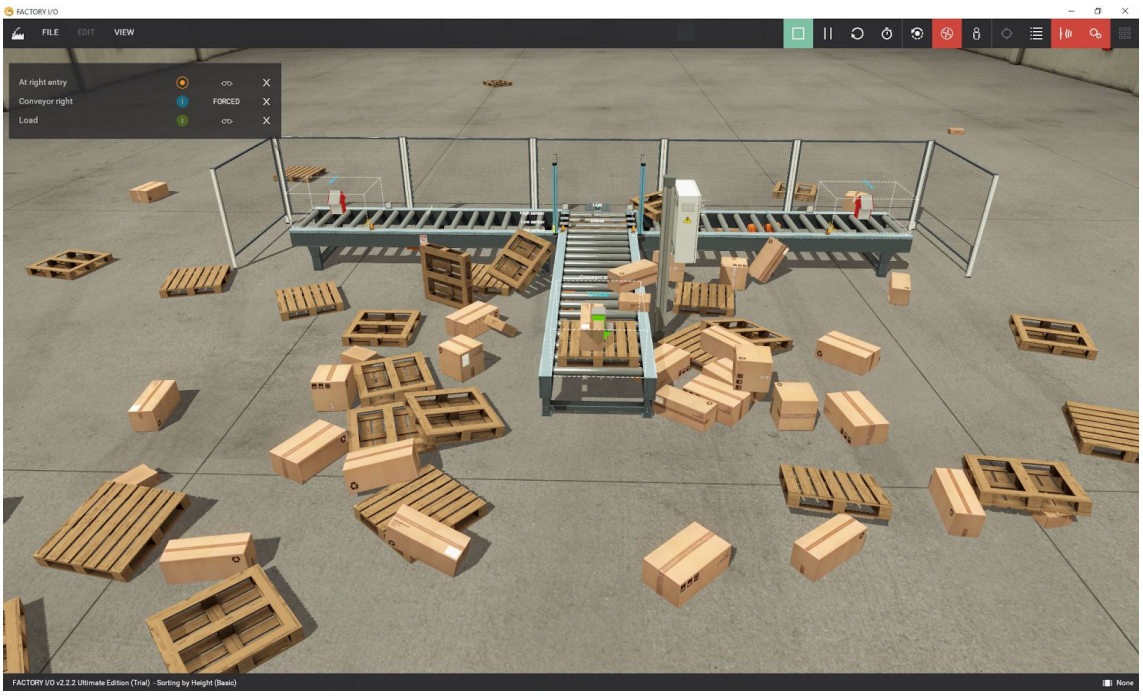

Figure 3

This was the result of our first testing phase of checking that Factory I/O. This test showed that Factory I/O can use PLC's to control a system and that when programmed incorrectly have a visual

display that something went wrong. This was one of the reasons we chose Factory I/O for feedback and this test proved it will work.

3.5 Functional Testing

Functional testing will vary for the different parts of our system. The front end UI will be able to use the automated testing mentioned above, checking each available option on the page does what is expected. In some scenarios, some of these options on the UI will only be able to be seen in the Factory I/O software or from the server command line. For these tests, a manual check will have to be made to guarantee the test was a success. The functional testing for Factory I/O is different because it depends less on user input and more on machine stability. The user should only have to interact with the Factory I/O software when they want to look at the models. The biggest issue is the stability of the software, so running the servers with different scenarios for long periods of time is the most accurate rigorous testing that can be performed.

We will also be performing a User Test case of a Mock CDC. This will take a group of users and allow them to run through our product and make sure that the system is usable to students. With these tests we plan on testing the usability of our product, and whether or not students with no experience with a PLC are able to understand the system enough to secure it. We will also be testing functionality of the vulnerabilities and ease of our user interface. We want the interface to be easy enough to understand that students spend more time securing then figuring the system out. This test will require a Red team which will probably be us, a blue team of users, and a fully operational product to allow the blue team to try and secure and find any bugs we may have missed.

3.6 Non-Functional Testing

One of the main components for our project is developing software that is intentionally vulnerable. Normally, one would test software to ensure that it is free from security vulnerabilities, but in our case we need to test to make sure that the vulnerabilities do work. Testing the vulnerabilities will involve manual testing for the most part, but some automation can be done to verify that the vulnerabilities still work as mentioned in section 3.3.

Another aspect of non-functional that we need to consider is performance testing. We will be running a fairly graphics intensive program on our Windows Servers and therefore, we need to ensure that the system is able to properly render the graphics without many performance issues. This type of testing may involve playing with the server settings to ensure a proper amount of resources are allocated to the Windows machines.

3.7 Modeling and Simulation

The teams model for the system can be seen in Figure 3, where the systems are designated and their software details specified. Because our project is software based, the chore of prototyping is greatly simplified. Being able to make any changes to our system and then revert them easily has been and will continue to be extremely helpful in reaching our end product.

An aspect of our project we cannot model so easily is the competition. Our team has become familiar with our project in such a way that manually re-creating the environment does not truly represent testing. To counter this, we aim to deliver our finished product to friends and

acquaintances who are totally unfamiliar with PLC's and the Factory I/O software. We expect the multiple amateur users to simulate the issues others may have in the competition.

3.8 Implementation Issues and Challenges

Through our design process we have ran into a few implementation challenges using the Factory I/O and working with it. The first challenge was that OpenPLC is not the most used product and does not have many example of usages. This has led us to have to do large amounts of exploring and implementing of our own. The second challenge we have faced is working with Factory I/O and OpenPLC together. Factory I/O does have an API and does have a soft PLC implementation, but the documentation on both of these is lacking and has required us to explore the implementations we are using without the use of examples.

Closing Material

4.1 Conclusion

Our team has accomplished a significant amount of work so far. We discovered many uses of PLCs in cyber-physical systems and identified the most scalable option. Through discussions with our advisors, we have determined the most desirable option for teaching security in CDCs: simulating a factory floor. We have successfully set up the software so that the OpenPLC can communicate with the virtualized factory in a local environment, and we have put in the necessary requests to get this environment in ISEAGE, the CDC production environment. We are on track to have a test environment setup by winter break, so that we have all of spring semester to create scenarios and fill out vulnerabilities.

Our goal is to have a real-world example of cyber-physical security vulnerabilities to give students experience in defending them. We hope to expose many vulnerabilities for Blue Team to defend and learn about, and to provide ample hacking opportunity for Red Team to exploit. Additionally, we have and will speak to professionals in the industry to hear first-hand accounts of cyber attacks.

Our course of action is primarily focused on delivering a factory I/O/OpenPLC server interaction in ISEAGE by winter break. Therefore, we are in conversation with the ISEAGE staff to secure a physical server for our use. Until then, we are actively testing factory I/O and OpenPLC on our local machines. Additionally, we regularly meet to brainstorm scenarios, troubleshoot each other's code, and flush out our factory program.

4.2 References

ISEAGE Cyber Defense Competitions. (2017). *Home*. [online] Available at: https://cdc.iseage.org/ [Accessed 28 Nov. 2017].

PLC Training with FACTORY I/O. (2017). *Next-Gen PLC Training Software with FACTORY I/O*. [online] Available at: https://factoryio.com [Accessed 28 Nov. 2017].

The OpenPLC Project. (2017). *The OpenPLC Project*. [online] Available at: http://www.openplcproject.com [Accessed 28 Nov. 2017].