INDUSTRIAL CONTOL SYSTEMS

PROTECTING A PLC FROM CORRUPTION

PLC

Lab Description: This lab explores a few security issues related to the use of Programmable Logic Controllers (PLCs) in the management of Industrial Control Systems (ICS), or similar forms of infrastructure.

This lab manual is very brief. You should read this "Lab Description" section before starting the lab. The student is expected to be somewhat proficient in the Python programming language, and is expected to have performed the Labtainer "onewayhash" lab.

This PLC lab simulates the system illustrated in Figure 1. A PLC manages the water level of a creek-fed catfish pond, ensuring the water level does not exceed minimum and maximum limits.

You will interact with the sys_management system to load a program and configuration data into the PLC. You will also use the sys_management system to check the status of the PLC and to query which program and configuration data the PLC is running. You will not have direct access to the PLC subsystem, though you can interact with it via the sys_management computer.

A "Security Proxy" sits between the sys_management computer and the PLC. The vendor promised that this will prevent attacks on the PLC. You can draw your own conclusions about that claim You will interact with the Security Proxy in an attempt to make it useful. But first, start the lab as noted below (if you have not already done so).

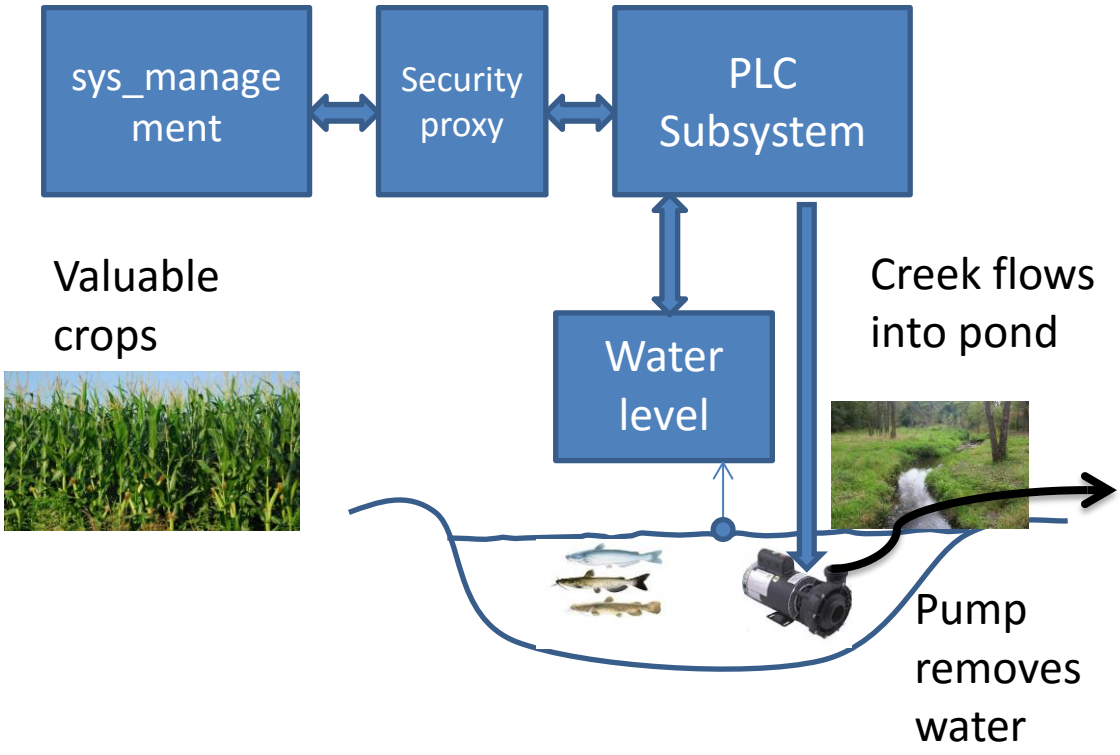

Figure 1: Farmer Jones Catfish Pond's Critical Infrastructure

Lab Environment:

The lab is started from the Labtainer workspace directory on your Docker-enabled host, e.g., a Linux VM. From there, issue the command:

```
./start.py plc
```

The resulting virtual terminals will include:

- A display of the status of the fish pond level, titled "Physical_World".
- A bash shell on the sys_management computer.
- A bash shell on the Security Proxy, titled "ubuntu@proxy".
- A display of the Security Proxy log file titled "PROXY_LOG".

NOTE: When the lab starts, observe the status window. The PLC is initially disabled, and thus the pump does not run and the water rises.

You can initialize the PLC from the sys_management window using:

```
./manage_plc.py load plc config.txt
```

The "plc" parameter is the name of the plc program file in your home directory. The "config.txt" is a configuration file in your home directory. This operation will initialize the PLC, leading to the pump to run.

The configuration file directs the PLC to keep the pond level between 20 and 30 feet. Just watch what happens over the course of about a minute.

After you've watched the status window for a full cycle of disaster, poke around a bit.

Hints:

- Use "./stop.py plc" and "./start.py plc" from your Linux host to stop and restart the lab -- this is the best way to restart the lab or reset the PLC if it becomes corrupt. Any files saved on the components will be preserved.
- The manage_plc.py tool lets your retrieve the code/data from the PLC. Are those the files you loaded?
- The sys_management computer includes the openssl utility that you used in the onewayhash lab, might that help determine if the files are the same?
- Could the Security Proxy be modified to avoid sending bad files to the PLC?

NOTE: The solution must use the manage_plc.py as-is. Modifying the code will void the warranty offered by the PLC system vendor!

Lab Files that are Needed: All of the files necessary for the lab are within the Labtainer components. Labtainers is retrieved from https://my.nps.edu/web/cisr/labtainers

LAB EXERCISE/STEP 1

Alter the proxy.py program on the Security Proxy computer to prevent exploitation of the PLC. You are not expected to make changes to the sys_management system, though you are free to explore it. However, credit will only be given if changes to the proxy.py mitigate the attack.

WHAT TO SUBMIT

When you have completed the lab, use "stop.py plc" to stop the lab, and provide the resulting zip file to your instructor.