

LAB 2: PERFORMING ATTACKS AND ADDING SECURITY MECHANISMS TO SCADA CONTROL SYSTEMS

Estimated Time: 1 hour and 40 minutes

Purpose: The purpose of this Lab exercise is to teach students how to create attacks on the control system and then how to create a defense mechanism by implementing iptables on the PLC Network.

Objective: Students will use Radzio! to perform a series of injection attacks on the PLC. These attacks should disrupt the normal behavior of the system. Additionally, students will write iptables rules to implement an access control list on the SCADA network traffic and therefore prevent the attacks from working.

Lab Setup and Requirements: To begin this lab, you will need to have VirtualBox and the Water Tank simulation running. Iptables will be configured and run on the Water Tank PLC.

EXERCISE #1 – INJECTION ATTACK ON THE LECTURER SIMULATION

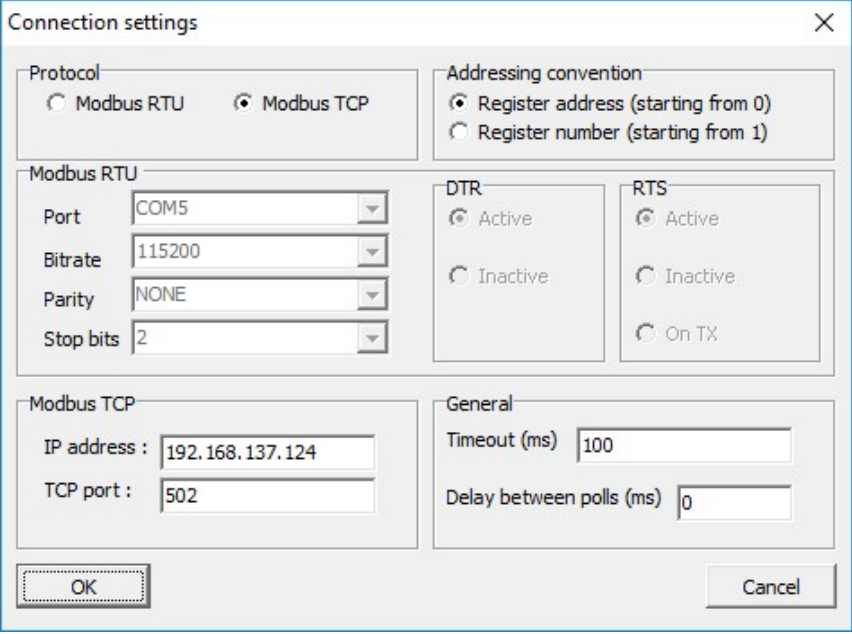
On this exercise students will perform an injection attack by fabricating messages with different settings and sending them to the target PLC. Students will use the Radzio! software to fabricate the messages.

The target will be the water tank PLC running on the lecturer's computer. After successfully attacking the lecturer simulation, a security mechanism will be put in place by the lecturer, and then the students will be asked to repeat the attack.

1. Download Radzio! from [here](#) and extract the contents of the zip file on a folder.

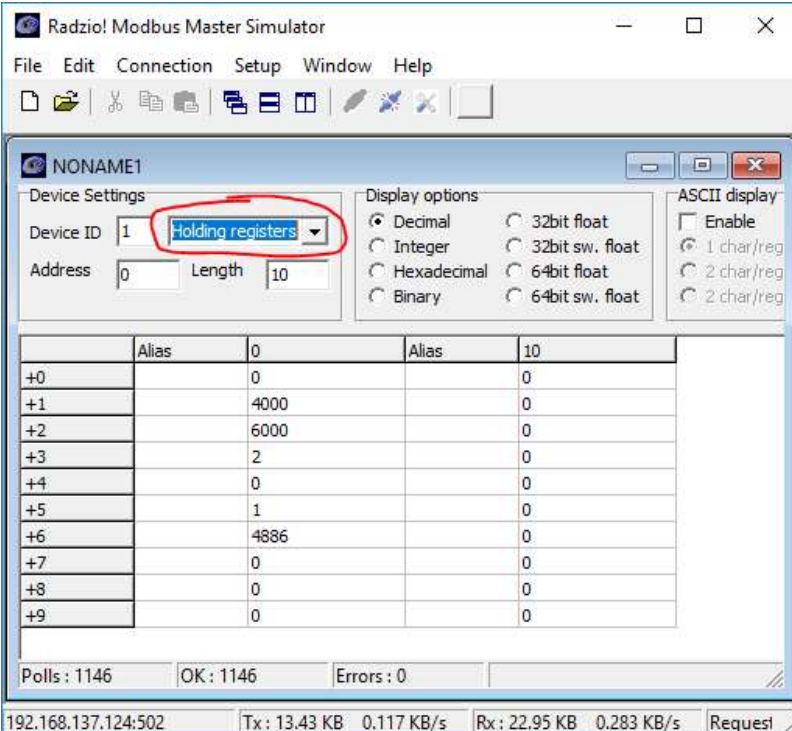


- Open RMMS.exe. On the main window, go to Connection->Settings. Select Modbus TCP under "Protocol", Register address starting from 0 under "Addressing convention", and type the target PLC address on "IP address: " field. Also, make sure that the TCP port is 502.



The "Connection settings" dialog box is shown. It has a "Protocol" section with "Modbus TCP" selected. The "Addressing convention" section has "Register address (starting from 0)" selected. The "Modbus TCP" section has "IP address: 192.168.137.124" and "TCP port: 502". The "General" section has "Timeout (ms): 100" and "Delay between polls (ms): 0". There are "OK" and "Cancel" buttons at the bottom.

- Click on File->New and Connection->Connect. On the new spreadsheet that appears, select Holding registers to view the PLC memory data



The "Radzio! Modbus Master Simulator" window is shown. The "Device Settings" section has "Device ID: 1" and "Address: 0" with a length of "10". The "Display options" section has "Holding registers" selected in a dropdown menu. The "ASCII display" section has "Enable" checked. The main area shows a table of PLC memory data.

	Alias	0	Alias	10
+0		0		0
+1		4000		0
+2		6000		0
+3		2		0
+4		0		0
+5		1		0
+6		4886		0
+7		0		0
+8		0		0
+9		0		0

At the bottom, there is a status bar showing "Polls: 1146", "OK: 1146", "Errors: 0", and a "Request" button.

4. The number on the second line of the spreadsheet (+1) has the min setpoint multiplied by 100: $4000 = 40\%$. Similarly, the third line (+2) has the max setpoint multiplied by 100: $6000 = 60\%$. Change both settings by double-clicking on each line and inserting a new value.
5. After successfully attacking the lecturer's simulation, disconnect by going to Connection->disconnect. Wait until the instructor has put the defense mechanism in place and then try the attack again by clicking on Connection->connect
6. Were you able to connect to the target PLC?

EXERCISE #2 - START SCADA LAB ENVIRONMENT

1. To start the virtual machine in the VirtualBox Manager, select the scadalab VM, right-click and select Start>Normal Start. Login using the credentials provided in Lab 1 (username:ccre, password:ccre).
2. Open Terminal by going to Applications -> Terminal Emulator
3. Navigate to the scripts folder with the command:

```
cd /home/ccre/scadalab/scripts
```

4. Run the script to configure the network with the command:

```
./netstart.sh
```

If it asks for a password, type: ccre

5. Start the Water Tank simulation with the command:

```
./watertank.sh
```

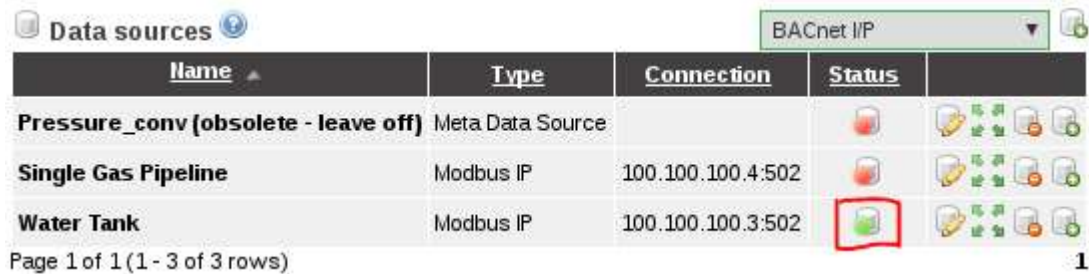
6. Launch the water tank HMI by opening the internet browser (Applications > Web Browser) and navigate to:

100.100.100.2:8080/ScadaBR

Login to ScadaBR using username:admin, password:admin



- Click on Data Sources on the top menu and then enable water tank data sources to allow ScadaBR to pull data from OpenPLC.



- Click on Graphical Views, select the Water Tank HMI from the drop down menu.



- Verify the levels are changing on the Water Tank HMI.

EXERCISE #3 – PREPARING RADZIO! FOR THE ATTACKS

On this exercise, students will prepare Radzio! Modbus to inject messages on the PLC running the Water Tank simulation

Section 1: Starting Radzio! Modbus

- Open Terminal by going to Applications -> Terminal Emulator
- Navigate to Radzio! folder with the command:

```
cd /home/ccre/scadalab/lab2/Radzio
```

- Start Radzio! with the command:

```
wine RMMS.exe
```

Section 2: Configuring Radzio! and connecting to the PLC

1. On the main window, go to Connection->Settings. Select Modbus TCP under "Protocol", Register address starting from 0 under "Addressing convention", and type 100.100.100.3 on the "IP address: " field, which is the target's IP address. Also, make sure that the TCP port is 502.
2. Click on File->New and Connection->Connect. On the new spreadsheet that appears, select Holding registers to view the PLC memory data, and make sure that the "Display options" is set to "Integer"

By following these steps, you now can see the memory data from the PLC. However, it can be a little hard to change settings from this window because we currently don't know what each number means. Section 3 will guide you to open the PLC program on the PLCOpen Editor to then identify where each variable is located in the PLC memory.

Section 3: Finding variables' location on the PLC memory

1. Open Terminal by going to Applications -> Terminal Emulator
2. Navigate to PLCOpen Editor folder with the command:

```
cd /home/ccre/scadalab/lab1/editor
```

3. Start the editor with the command:

```
python PLCOpenEditor.py
```

4. In the PLCOpen Editor, select File>Open. Navigate to ccre/scadalab/lab1 folder (you can find the ccre folder on the left sidebar). Double click the Water_Tower.xml file in the Lab 1 Directory.
5. Double-click on "My Program" and look at the variables table on the top of the screen



The screenshot shows the PLCOpen Editor window with a tab titled 'My Program'. Below the tab is a 'Description:' field and a 'Class Filter:' dropdown set to 'All'. The main area contains a table with the following data:

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	Level	Local	INT	%IW0			
2	Level_filtered	Local	INT	%QW6			
3	High_Alarm	Local	BOOL	%QX0.5			
4	Low_Alarm	Local	BOOL	%QX0.6			
5	min_sp	Local	INT	%QW1			

6. Observe the 5th column called "Location". All variables stored in the PLC data memory are located at %QWn, where n is a number from 0 to 1024. Write down the name and the location (the n position) of each variable located at %QWn



Section 3: Identifying variables on the PLC memory

1. Go back to Radzio! Modbus software.
2. Fill out the Holding Registers table with the information you collected from the previous section. The row +0 should contain the variable on the PLC program located at %QW0, the row +1 should contain the variable located at %QW1, and so on...
3. You should end up with a table like this one

The screenshot shows the Radzio! Modbus software interface. The title bar is 'NONAME1'. The 'Device Settings' section has 'Device ID' set to 1, 'Holding registers' selected in the dropdown, 'Address' set to 0, and 'Length' set to 10. The 'Display options' section has 'Decimal' selected. The 'ASCII display' section has 'Enable' checked. The 'Holding Registers' table is displayed with the following data:

	Alias	0	Alias	10
+0		0		0
+1	min_sp	4000		0
+2	max_sp	6000		0
+3	mode_register	2		0
+4	pump_register	0		0
+5	valve_register	1		0
+6	Level_filtered	5488		0
+7		0		0
+8		0		0
+9		0		0

At the bottom, the status bar shows 'Polls : 2159', 'OK : 2159', and 'Errors : 0'.

Now the information you see makes more sense. You should now be ready to start the attacks.

EXERCISE #4 – INJECTION ATTACKS

On this exercise, students will use Radzio! Modbus to inject messages on the PLC running the Water Tank simulation

Attack 1: Damaging the water pump

1. Set the min_sp bigger than the max_sp
2. On this setting, the system will turn the pump on and off frequently, which may stress the water pump.
3. To verify the pump controls, click on File->New and select Coil status. The pump status is on row +0.

Attack 2: Emptying the tank

1. Set min_sp to -5
2. On this setting, the system will empty the entire tank and the pump will never be turned on
3. You can verify the water level on the tank by looking at row +6: Level_filtered. You can also check the status of the system through the HMI on the Web Browser.

Attack 3: Overflowing the tank

1. Set mode_register to 1
2. Set pump_register to 1
3. Set valve_register to 0
4. On this setting, the pump will be locked on, never turning off. Also, the valve will be closed, preventing the water in the tank to flow back to the reservoir. This is the worst scenario possible!
5. In a few seconds, you can verify that the water level goes above 100%. You can check the water level on the tank by looking at row +6: Level_filtered. You can also check the status of the system through the HMI on the Web Browser.

EXERCISE #5 – SETTING UP DEFENSES

Iptables is a rule-based firewall software. The user can define which machine is allowed to communicate on the network by using a set of rules. On this exercise, students will write iptables rules that will prevent unauthorized users to connect to the PLC.

Section 1: Writing iptables rules

1. On Radzio! go to Connection->Disconnect, and then close the program
2. Open Terminal by going to Applications -> Terminal Emulator
3. Log in the PLC shell by typing the command:

```
sudo docker exec -it plc0 bash
```

4. Create an iptables rules file with the command:

```
nano /etc/iptables.test.rules
```



5. This command will open an editor window showing a blank file. Type the following inside the editor:

```
*filter

#Drop everything but our output to internet
-P FORWARD DROP
-P INPUT DROP
-P OUTPUT ACCEPT

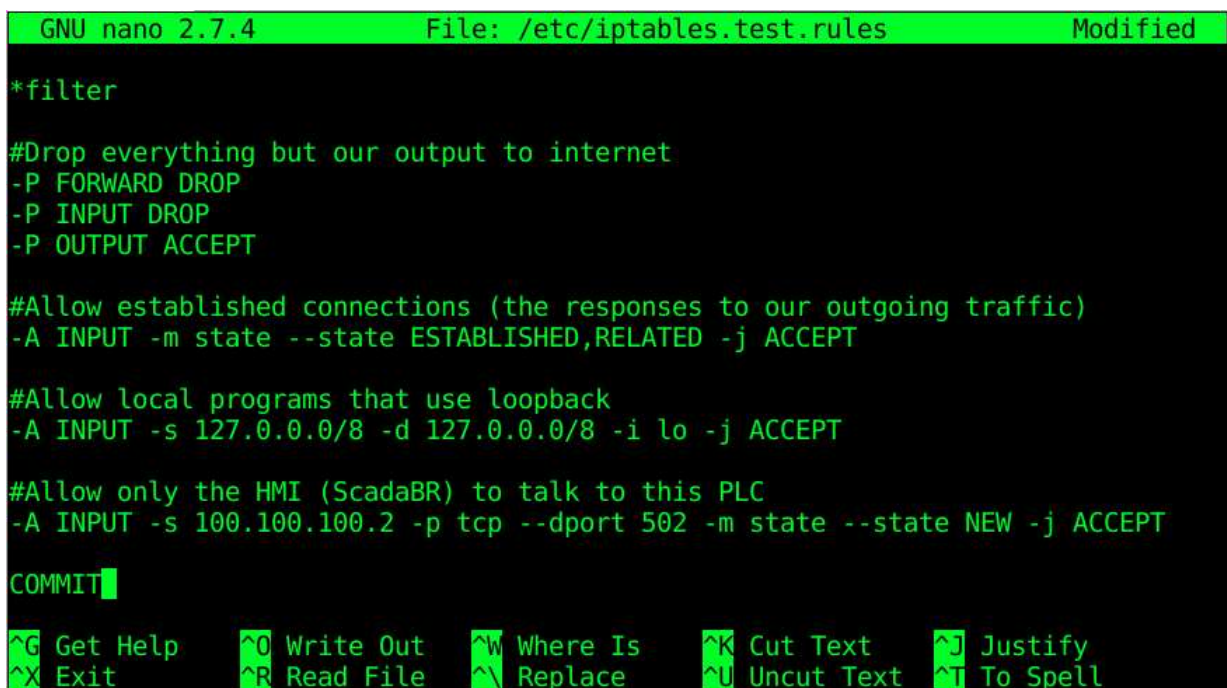
#Allow established connections (the responses to our outgoing traffic)
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

#Allow local programs that use loopback
-A INPUT -s 127.0.0.0/8 -d 127.0.0.0/8 -i lo -j ACCEPT

#Allow only the HMI (ScadaBR) to talk to this PLC
-A INPUT -s 100.100.100.2 -p tcp --dport 502 -m state --state NEW -j ACCEPT

COMMIT
```

Lines starting with the # character are comments and can be omitted. This file set rules to block all incoming connections (-P INPUT DROP), except the connections coming from the HMI (-A INPUT -s 100.100.100.2 ...)



```
GNU nano 2.7.4      File: /etc/iptables.test.rules      Modified

*filter

#Drop everything but our output to internet
-P FORWARD DROP
-P INPUT DROP
-P OUTPUT ACCEPT

#Allow established connections (the responses to our outgoing traffic)
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

#Allow local programs that use loopback
-A INPUT -s 127.0.0.0/8 -d 127.0.0.0/8 -i lo -j ACCEPT

#Allow only the HMI (ScadaBR) to talk to this PLC
-A INPUT -s 100.100.100.2 -p tcp --dport 502 -m state --state NEW -j ACCEPT

COMMIT

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^N Replace   ^U Uncut Text ^I To Spell
```

6. Save the file and exit by typing Ctrl+X and then Y

Section 2: Applying the new rules and testing the configuration

1. Back to the terminal, type the following command to apply the recently created rules:

```
iptables-restore < /etc/iptables.test.rules
```

2. Verify if the new rules were applied with the command:

```
iptables -L
```

```
root@bf32a4dfac16:/OpenPLC_v2# iptables -L
Chain INPUT (policy DROP)
target    prot opt source                destination            state RELATED,ESTABLISHED
ACCEPT    all  --  anywhere              anywhere
ACCEPT    all  --  loopback/8            loopback/8
ACCEPT    tcp  --  HMI.plcnet            anywhere               tcp dpt:502 state NEW

Chain FORWARD (policy DROP)
target    prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
```

With this command, we can verify that the default policy for incoming connections (Chain INPUT) is DROP, which means that it will reject all incoming connections. The exceptions to this rule are the three listed under this section. The first exception states that it should accept all incoming connections that are responses from requests (state RELATED). The second exception states that it should accept all connections coming from loopback. These are internal connections and therefore are harmless. The third exception states that any connection from the HMI should be accepted.

3. Verify if the HMI can still communicate with the PLC by opening the Web Browser. Change a few settings (min and max setpoints) to make sure that the system is working properly.
4. Repeat Exercise #3 (Sections 1 and 2) and verify if you can still have access to the PLC

ACKNOWLEDGEMENTS

This lab was developed at the University of Alabama in Huntsville by Stefanie Smith, Ben McGee, Thiago Alves, Joseph Lee, and Tommy Morris.

OpenPLC is a completely open programmable logic controller with development environment, human machine interface, programmable logic controller source code, and reference hardware available at <http://www.openplcproject.com/>. The OpenPLC project was founded by Thiago Alves of the University of Alabama in Huntsville.

The Simulink models, human machine interface implementation, and ladder logic programming used for the gas pipeline and water storage tank test beds used for this laboratory exercise are the copyrighted property of the University of Alabama in Huntsville.

