Programmable Logic Controllers (PLCs)
Lab #2 Intro to PLC Manipulation

Intro:

In this lab you will abuse features provided by the PLC to achieve undesirable effects.

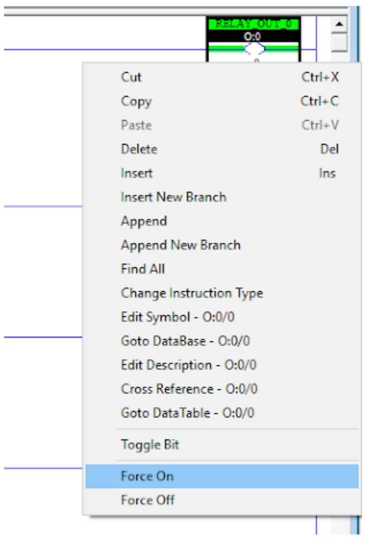
Setup

- For this lab, you will use:
  - The ICS kit from Lab 1
  - The ICS VM from Lab 1
  - A VM of Kali, run on the same host as the ICS VM
    - You can use a new or existing instance of Kali. Up to you.

- Network setup
  - Ensure the ICS VM and the PLC are running, and one of the programs you created in Lab 1 is operating in Run mode.
  - Fire up the Kali VM
  - Ensure the VMWare network settings for this VM are set to Bridged Mode, bridged to the same interface that the PLC is plugged into, and that the IP address of the VM is set to the same subnet as the VM and PLC from Labs 0 and 1 **(verify using ping).**

Sniff traffic

- The objective in this part of the lab is to capture the packets between the ICS VM and the PLC when performing a specific function
- To accomplish this, you are going use Wireshark to capture four specific events when the RSLogix software is "talking" to the PLC
  - **Place the PLC into "Run" mode**
    - Start with the PLC in "Program" mode and switch to "Run" mode
  - **Place the PLC into "Program" mode**
    - Start with the PLC in "Run" mode and switch to "Program" mode
  - **Force the contents of a variable on the PLC to low**
    - With the PLC in "Run" mode, "Force" the value of an digital output used in your program (e.g. O:0/0), by selecting in RSLogix Project/Data Files/Output and then clicking on the Forces button.
    - Double-click on the output variable you want to force a change to, set the value to 0, and press "Enter".
    - As soon as you press "Enter", the variable on the PLC is updated.
    - You can also right-click on an output as shown here.



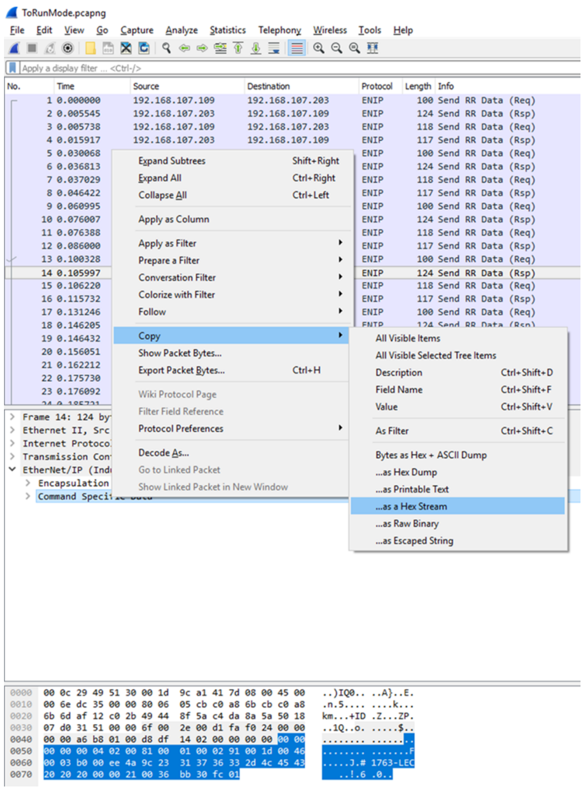
  - **Force the contents of a variable on the PLC to high**

    - With the PLC in “Run” mode, “Force” the value of an output variable used in your program (e.g. O:0.0/0), by selecting in RSLogix Project/Data Files/Output and then clicking on the Forces button.
    - Double-click on the output variable you want to force a change to, set the value to 1, and press “Enter”.
    - As soon as you press “Enter”, the variable on the PLC is updated. You can verify by checking the lights in the kit to see if the expected outcome was achieved.
- **NOTE: When the PLC is in Run mode, it is constantly talking to the RSLogix software, generating a significant amount of traffic. To minimize the amount of packets we need to analyze, start Wireshark sniffing on the same interface as the ICS VM right before performing the planned action on the PLC, and then stop Wireshark immediately after performing the planned action. Save the packet capture and name it appropriately for later analysis.**

Analyze Network Traffic

- Your goal in this section is to identify the packet in Wireshark that corresponds to the actions taken in RSLogix.
- Open the previously saved Wireshark capture corresponding to the action of interest.
- Since you are looking for a packet sent from RSLogix to the PLC, configure an appropriate Wireshark display filter (e.g., ip.src==192.168.107.122).
- The traffic generated by being in Online Mode will repeat often, creating a lot of repetitive noise.
- If you captured the traffic correctly, the command from RSLogix to the PLC occurs in just one packet.
- **Your task is to find that one packet. One packet will not be like the others. Use your imagination to find it.**

- Once you've found the correct packet, you will need to copy the "Hex Stream" data.
  - If Wireshark shows ENIP as the protocol (like the screenshot below), then find:
    Ethernet/IP (Industrial Protocol) >> Command Specific Data
  - If Wireshark shows CIP as the protocol, then find:
    CIP Class Generic >> Command Specific Data
  - Right click on the "**Command Specific Data**" line and select:
    Copy >> Bytes as a Hex Stream



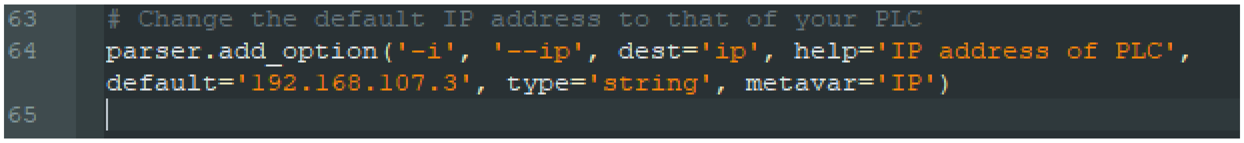
```
No.  Time      Source           Destination      Protocol Length Info
   1 0.000000  192.168.107.109  192.168.107.203  ENIP     100 Send RR Data (Req)
   2 0.005545  192.168.107.203  192.168.107.109  ENIP     124 Send RR Data (Rsp)
   3 0.005738  192.168.107.109  192.168.107.203  ENIP     118 Send RR Data (Req)
   4 0.015917  192.168.107.203  192.168.107.109  ENIP     117 Send RR Data (Rsp)
   5 0.030068                                    ENIP     100 Send RR Data (Req)
   6 0.036813                                    ENIP     124 Send RR Data (Rsp)
   7 0.037029                                    ENIP     118 Send RR Data (Req)
   8 0.046422                                    ENIP     117 Send RR Data (Rsp)
   9 0.060995                                    ENIP     100 Send RR Data (Req)
  10 0.076007                                    ENIP     124 Send RR Data (Rsp)
  11 0.076388                                    ENIP     118 Send RR Data (Req)
  12 0.086000                                    ENIP     117 Send RR Data (Rsp)
  13 0.100328                                    ENIP     100 Send RR Data (Req)
  14 0.105997                                    ENIP     124 Send RR Data (Rsp)
  15 0.106220                                    ENIP     118 Send RR Data (Req)
  16 0.115732                                    ENIP     117 Send RR Data (Rsp)
  17 0.131246                                    ENIP     100 Send RR Data (Req)
  18 0.146205                                    ENIP     124 Send RR Data (Rsp)
  19 0.146432
  20 0.156051
  21 0.162212
  22 0.175730
  23 0.176092

> Frame 14: 124 by
> Ethernet II, Src
> Internet Protoco
> Transmission Con
v EtherNet/IP (Ind
  > Encapsulation
  > Command Speci

0000  00 0c 29 49 51 30 00 1d  9c a1 41 7d 08 00 45 00   ..)IQ0.. ..A}..E.
0010  00 6e dc 35 00 00 80 06  05 cb c0 a8 6b cb c0 a8   .n.5.... ....k...
0020  6b 6d af 12 c0 2b 49 44  8f 5a c4 da 8a 5a 50 18   km...+ID .Z...ZP.
0030  07 d0 31 51 00 00 6f 00  2e 00 d1 fa f0 24 00 00   ..1Q..o. .....$..
0040  00 00 a6 b8 01 00 d8 df  14 02 00 00 00 00 00 00   ........ ........
0050  00 00 00 04 02 00 81 00  01 00 02 91 00 1d 00 46   ........ .......F
0060  00 03 b0 00 ee 4a 9c 23  31 37 36 33 2d 4c 45 43   .....J.# 1763-LEC
0070  20 20 20 00 00 21 00 36  bb 30 fc 01                 ..!.6 .0..
```

- ○ Save the results into a document until you can place them into your script
- ○ You will paste the results from your network analysis in the attack script in the next step.

- Perform this action for each of the required four options.

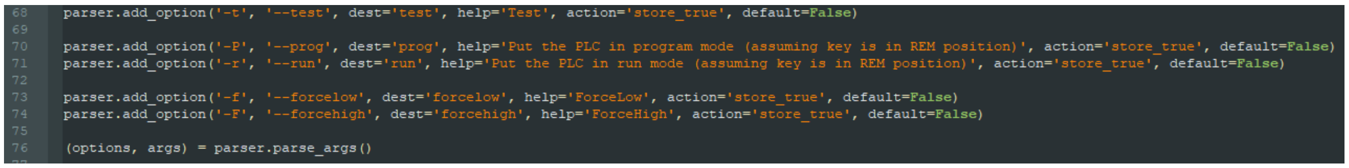
Write the Exploit

- You have been provided a Python script template for use in this section of the lab. **The script is in a folder on the file server (CIP_ML1100_exploit.py).**
- The first thing you must do is **change the default IP address in the script** to that of your PLC.



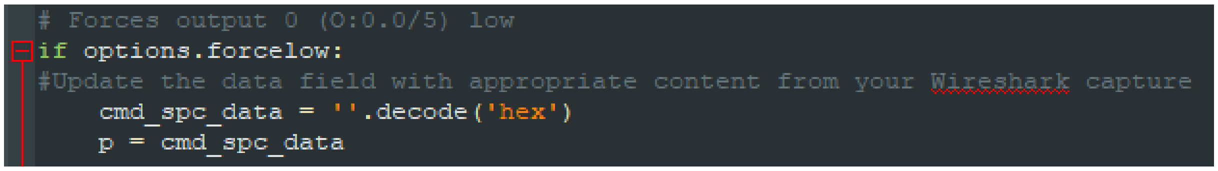
```python
# Change the default IP address to that of your PLC
parser.add_option('-i', '--ip', dest='ip', help='IP address of PLC',
default='192.168.107.3', type='string', metavar='IP')

```

- You will notice there are five options in the Python code. “options.test” is simply for querying the status of the PLC. You will not modify this option. **Use this option to check connectivity.**
- The remaining four options will be modified with data from your Wireshark capture of the matching events performed in RSLogix.



```python
parser.add_option('-t', '--test', dest='test', help='Test', action='store_true', default=False)

parser.add_option('-P', '--prog', dest='prog', help='Put the PLC in program mode (assuming key is in REM position)', action='store_true', default=False)
parser.add_option('-r', '--run', dest='run', help='Put the PLC in run mode (assuming key is in REM position)', action='store_true', default=False)

parser.add_option('-f', '--forcelow', dest='forcelow', help='ForceLow', action='store_true', default=False)
parser.add_option('-F', '--forcehigh', dest='forcehigh', help='ForceHigh', action='store_true', default=False)

(options, args) = parser.parse_args()
```

- Paste the results from your network analysis in the **cmd_spc_data** variable in each of the appropriate options.



```python
# Forces output 0 (O:0.0/5) low
if options.forcelow:
#Update the data field with appropriate content from your Wireshark capture
    cmd_spc_data = ''.decode('hex')
    p = cmd_spc_data
```

Execute the Attack

- Execute the attack by running the Python script with the appropriate option (e.g. Python ML1100_exploit.py -P)

    - Run all four versions of the PLC attack (Program, Run, OutputLow, OutputHigh)
- Explore RSLogix capabilities and identify another attack vector, where RSLogix is “talking” to the PLC and you have the ability to sniff the data packets.
- Create your own “option” in the Python script, comment the code appropriately, perform your custom attack against the PLC, and document your results.
    - Please be careful to not intentionally “brick” the PLC. They are not cheap.