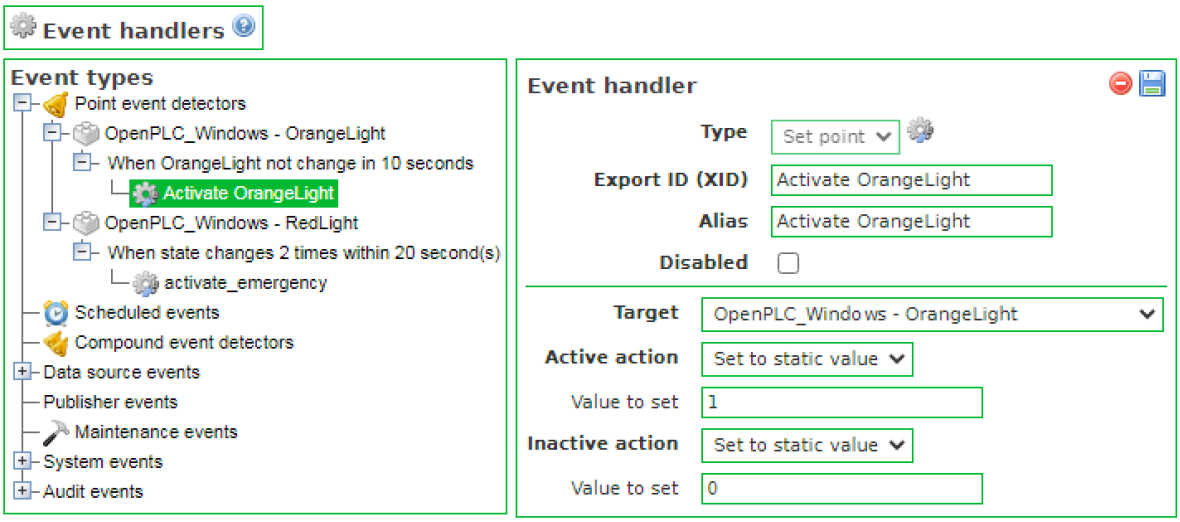
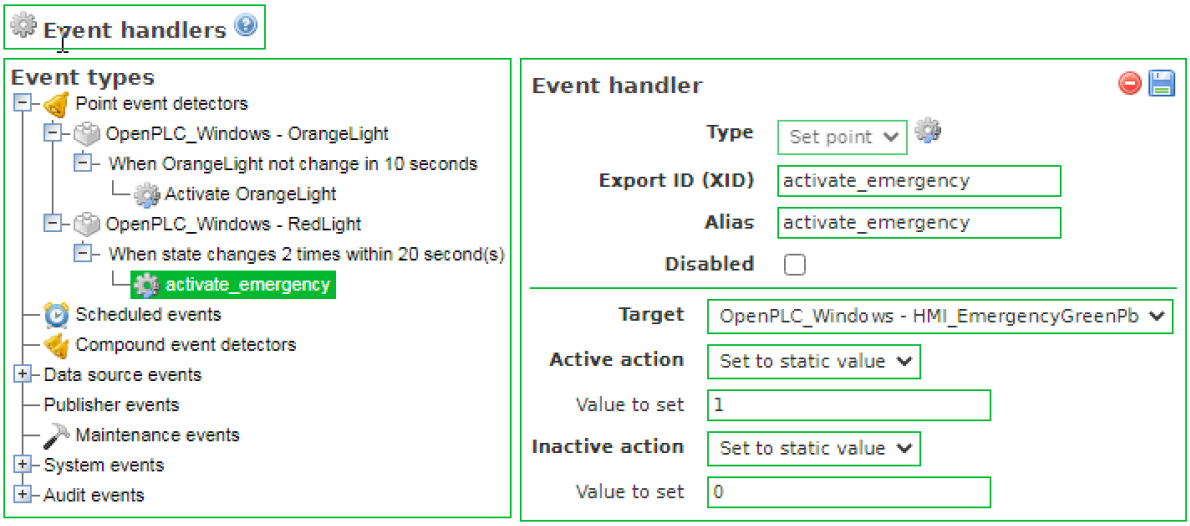
Produce equivalent data for multiple events

Here's how those events were set.

1. If RedLight changes twice in 20 seconds => Activate Emergency, when condition not met, Deactivate Emergency
2. If OrangeLight not change in 10 seconds => Make OrangeLight to 1, when condition not met, Make OrangeLight to 0





Running Timing and Frequency Script

```
..esktop/CS6727 (-zsh)  ⌘1   twang626@bird: ~/Desktop/11_1/multiple_events (-zsh)  ⌘2   ..tack Showcase (-zsh)  ⌘3
twang626@bird:~/Desktop/11_1/multiple_events$ python3 timing_and_frequency_analysis.py
Write_to_Write_Timing:
        Count: 276 in total
        Mean: 4335.96 milliseconds
        Standard Deviation: 4612.17 milliseconds
Read_to_Write_Timing:
        Count: 186 in total
        Mean: 97.54 milliseconds
        Standard Deviation: 91.79 milliseconds
Read_to_Read_Timing:
        Count: 1151 in total
        Mean: 1043.98 milliseconds
        Standard Deviation: 1179.39 milliseconds

Cycle Frequencies (Use a WRITE TO WRITE as a cycle)
        Cycle Count: 276 in total
        Frequency Sequences: [(1.0, 0.00062637), (0.2, 5.716e-05), (0.33333333, 6.964e-05), (1.0, 0.00059916), (0.16666
667, 0.00013057), (0.5, 0.00084746), (1.0, 0.00109769), (0.5, 0.00087222), (0.5, 0.00043057), (1.0, 0.00159109), (0.2,
0.00096479), (1.0, 0.00769231), (0.16666667, 0.00070249), (0.16666667, 0.00053865), (0.33333333, 0.00117647), (1.0, 0.0
0660066), (0.16666667, 0.00105988), (0.5, 0.00147493), (1.0, 0.04545455), (0.5, 0.00410678), (0.5, 0.00113507), (1.0, 0
.003861), (0.2, 0.00079334), (1.0, 0.00772201), (0.16666667, 0.00077369), (0.16666667, 0.00055066), (0.33333333, 0.0009
8765), (1.0, 0.00769231), (0.16666667, 0.0007874), (0.33333333, 0.00150943), (0.5, 0.00628931), (1.0, 0.00386847), (0.2
, 0.00072833), (1.0, 0.00766284), (0.16666667, 0.00052549), (0.16666667, 0.0006805), (0.33333333, 0.00136799), (1.0, 0.
00769231), (0.16666667, 0.00077399), (0.5, 0.00104112), (1.0, 0.04347826), (0.5, 0.00263505), (0.66666667, 0.00059827),
 (1.0, 0.00274348), (1.0, 0.5), (0.16666667, 0.00071403), (0.33333333, 0.00105485), (0.16666667, 0.00056545), (1.0, 0.0
0772201), (0.16666667, 0.0007734), (0.16666667, 0.00048112), (0.33333333, 0.00235849), (1.0, 0.00772201), (0.16666667,
0.00052562), (0.33333333, 0.00150943), (0.5, 0.00609756), (1.0, 0.00393701), (0.2, 0.00074156), (1.0, 0.00769231), (0.1
6666667, 0.00077399), (0.16666667, 0.00047642), (0.33333333, 0.00136893), (1.0, 0.00772201), (0.16666667, 0.00077399),
(0.33333333, 0.00105708), (0.5, 0.00205761), (1.0, 0.003861), (0.2, 0.00095602), (1.0, 0.00766284), (0.16666667, 0.0005
2562), (0.5, 0.00285714), (1.0, 0.04347826), (0.2, 0.00071659), (0.33333333, 0.00110926), (1.0, 0.00772201), (0.1666666
7, 0.0007734), (0.33333333, 0.00128949), (0.5, 0.0012945), (1.0, 0.00392927), (0.2, 0.00135501), (1.0, 0.00769231), (0.
33333333, 0.00047996), (0.5, 0.00253165), (1.0, 0.0625), (0.25, 0.00052507), (0.33333333, 0.00407332), (1.0, 0.00772201
), (0.16666667, 0.00077369), (0.33333333, 0.00094384), (0.5, 0.00205761), (1.0, 0.00407332), (0.2, 0.00072124), (1.0, 0

        Mean: (0.53482155, 0.01506487)
        Standard Deviation: (0.34558552, 0.07278189)
```

Statistics

All: 3,501,659 API calls
SCADABR APIs: 940,597 API calls
SCADABR and Dependencies APIs: 940,680 API calls
READ commands: 1,152 API calls
WRITE commands: 277 API calls

Injecting Attack

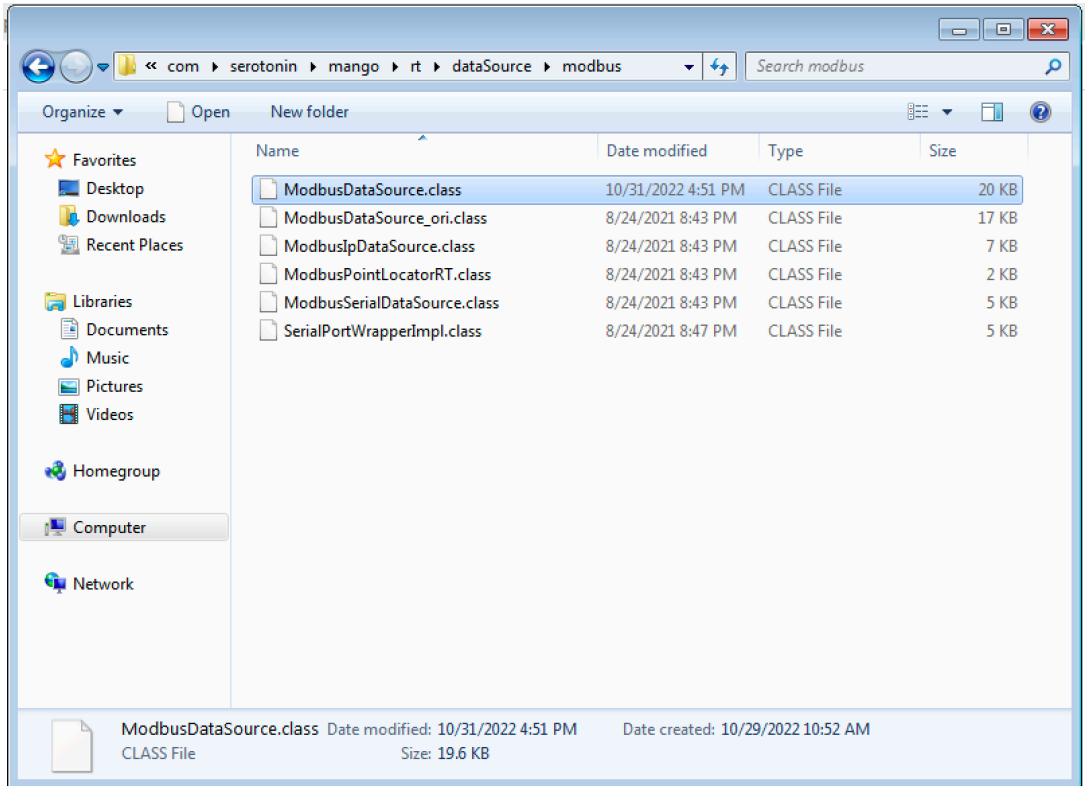
Target Java File: https://github.com/ScadaBR/ScadaBR/blob/master/src/com/serotonin/mango/rt/dataSource/modbus/ModbusDataSource.java

This file is responsible for reading Modbus data from OpenPLC, and there is a function inside to send Modbus data to OpenPLC.

Install Eclipse (version 2019-09) and JDK 8 to compile the ScadaBR project. It compiles .java file to Java bytecode .class file.
To perform the attack, I substitute the file C:\Program Files\ScadaBR\tomcat\webapps\ScadaBR\WEB-INF\classes\com\serotonin\mango\rt\dataSource\modbus\ModbusDataSource.class with our newly compiled ModbusDataSource.class.

Attack Condition

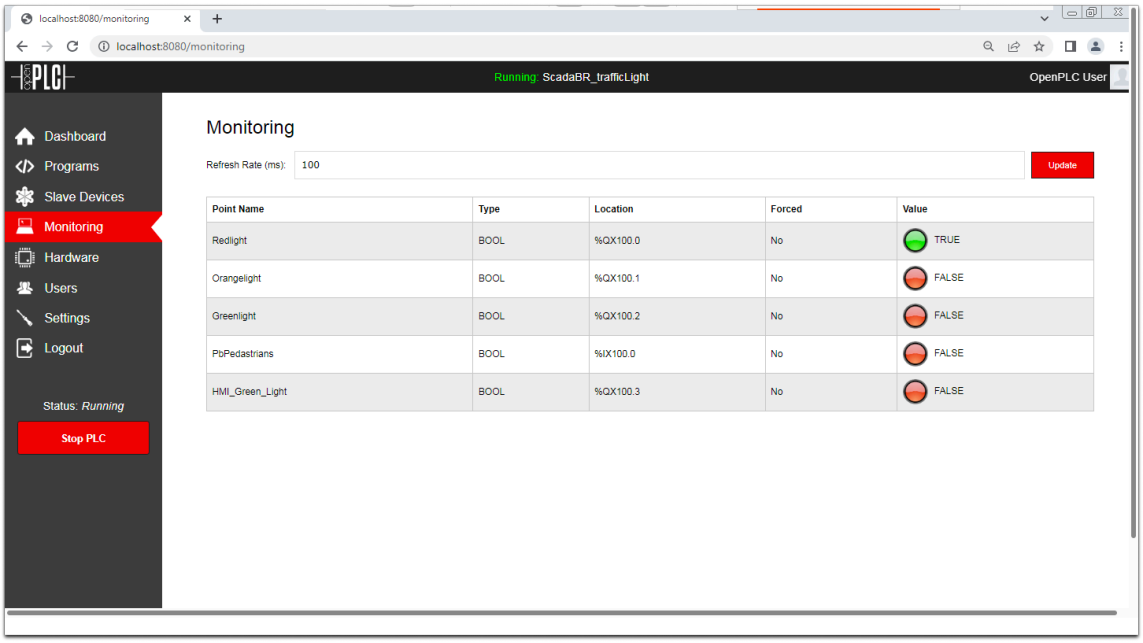
Condition1: GreenLight = 1 && Emergency = 1 -> WRITE VictimCoil = 1
Condition2: GreenLight = 1 && Emergency = 0 -> WRITE VictimCoil = 0



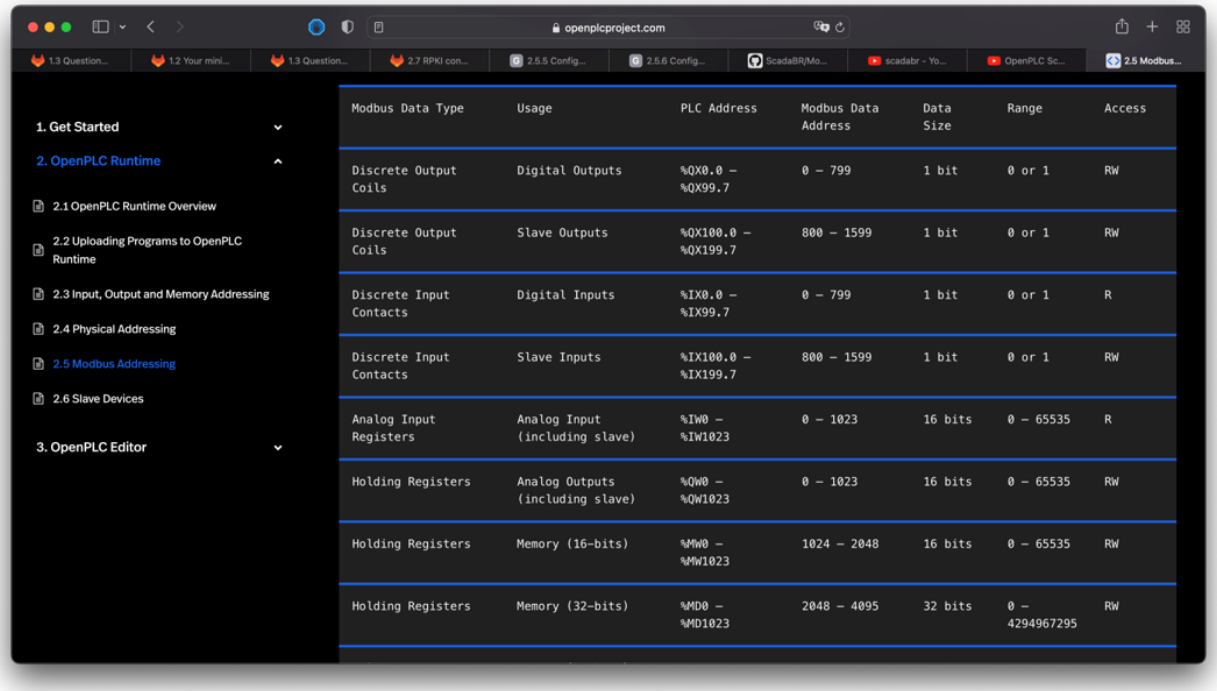
To create a Victim Coil

I created VictimCoil to show my attack indeed works. This VictimCoil data cannot be changed by the Traffic Light program on OpenPLC.

First, access OpenPLC Runtime, runs the PLC program, and goes to the “Monitoring” tab. There are Point Name (Coil Name) with their locations.

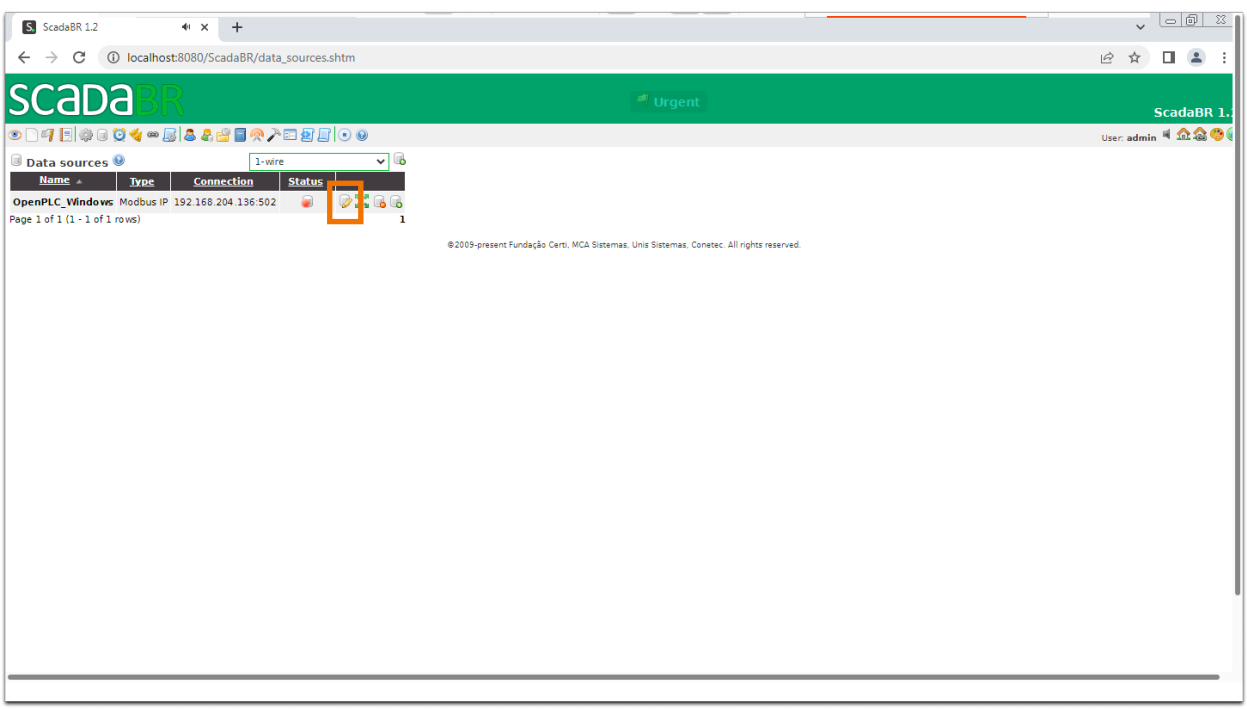


From the documentation in OpenPLC (https://openplcproject.com/docs/2-5-modbus-addressing/), it shows us how to map location in OpenPLC to ScadaBR. For instance, %QX100.0 is 800 in ScadaBR.

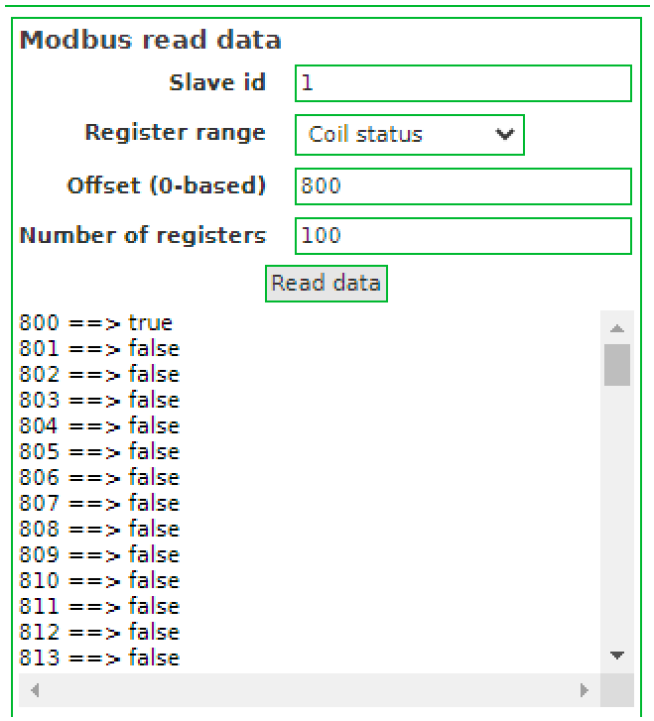


Modbus Data Type	Usage	PLC Address	Modbus Data Address	Data Size	Range	Access
Discrete Output Coils	Digital Outputs	%QX0.0 – %QX99.7	0 – 799	1 bit	0 or 1	RW
Discrete Output Coils	Slave Outputs	%QX100.0 – %QX199.7	800 – 1599	1 bit	0 or 1	RW
Discrete Input Contacts	Digital Inputs	%IX0.0 – %IX99.7	0 – 799	1 bit	0 or 1	R
Discrete Input Contacts	Slave Inputs	%IX100.0 – %IX199.7	800 – 1599	1 bit	0 or 1	RW
Analog Input Registers	Analog Input (including slave)	%IW0 – %IW1023	0 – 1023	16 bits	0 – 65535	R
Holding Registers	Analog Outputs (including slave)	%QW0 – %QW1023	0 – 1023	16 bits	0 – 65535	RW
Holding Registers	Memory (16-bits)	%MW0 – %MW1023	1024 – 2048	16 bits	0 – 65535	RW
Holding Registers	Memory (32-bits)	%MD0 – %MD1023	2048 – 4095	32 bits	0 – 4294967295	RW

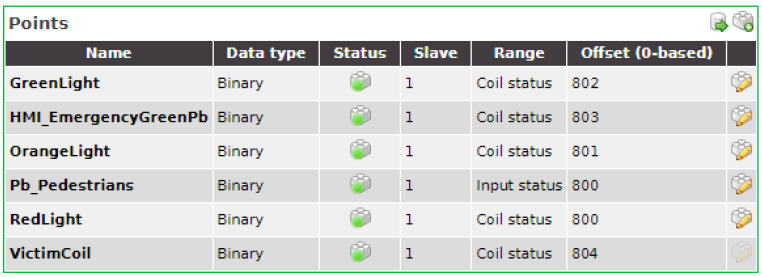
Go to ScadaBR, access Data Sources Tab. Click “Edit” in the orange box below.



There is a Modbus Read Section. Type “800” on the offset,
specify the number of registers, and click “Read data”. We can read data manually from OpenPLC.

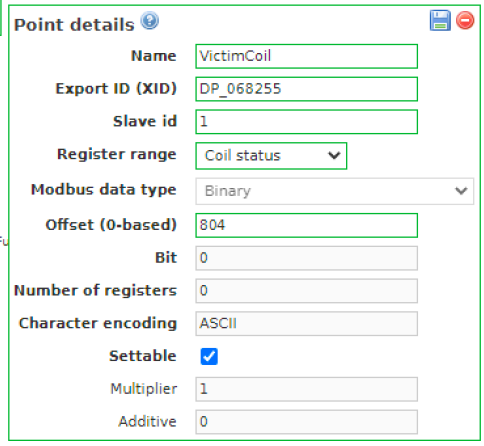


From OpenPLC, we have seen that %QX100.4 is not used, which maps to 804 in ScadaBR. We can create a VictimCoil at that register on the Point Section on the same page, and that is our VictimCoil. ScadaBR will read the point value from OpenPLC regularly.



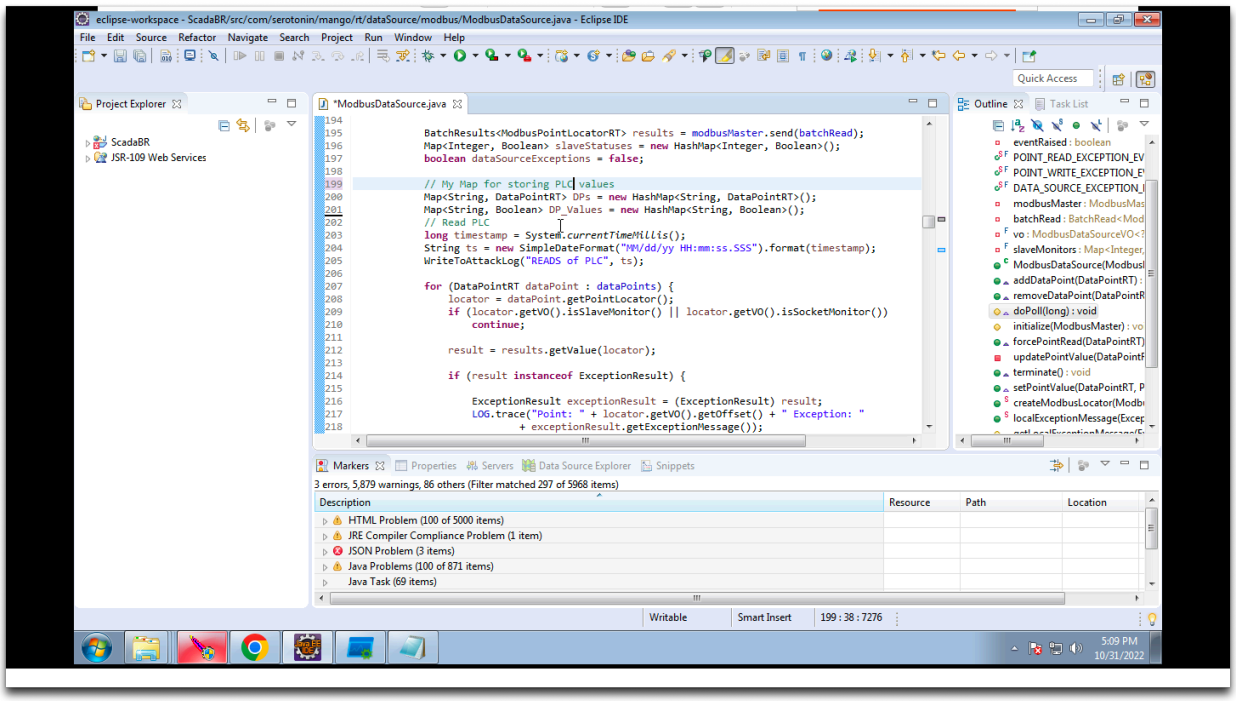
Points

Name	Data type	Status	Slave	Range	Offset (0-based)
GreenLight	Binary		1	Coil status	802
HMI_EmergencyGreenPb	Binary		1	Coil status	803
OrangeLight	Binary		1	Coil status	801
Pb_Pedestrians	Binary		1	Input status	800
RedLight	Binary		1	Coil status	800
VictimCoil	Binary		1	Coil status	804

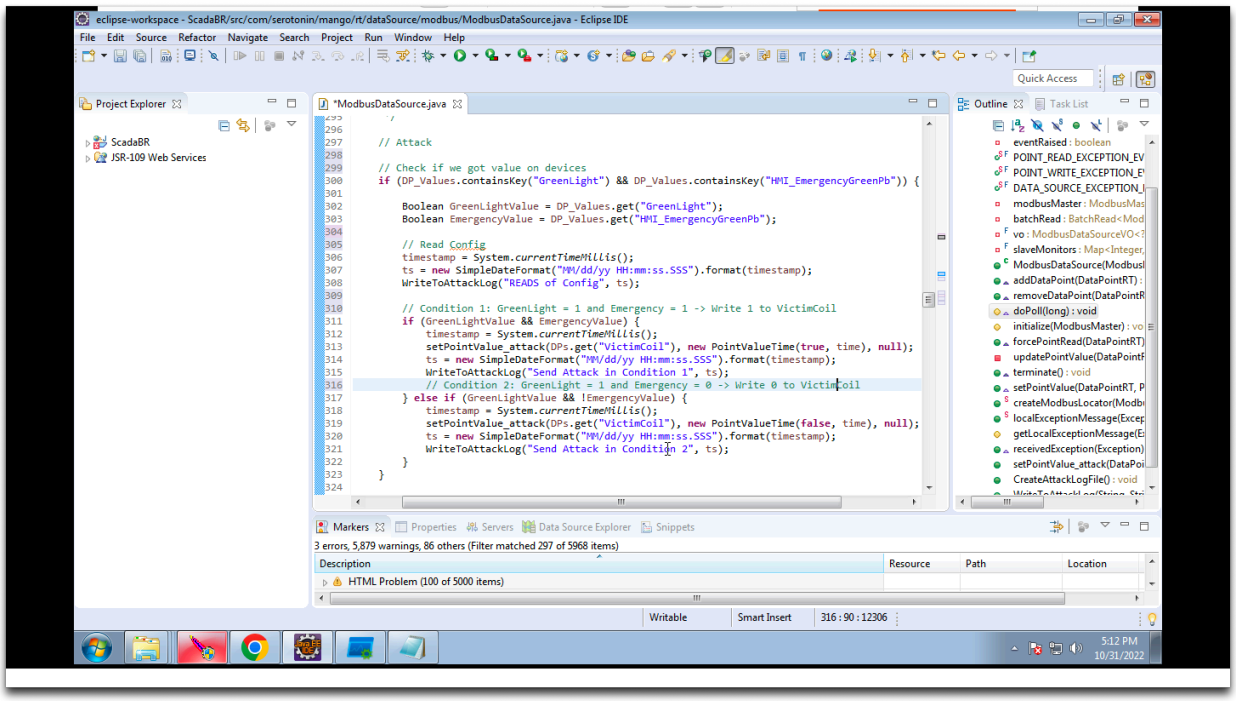


Some Code Snippets

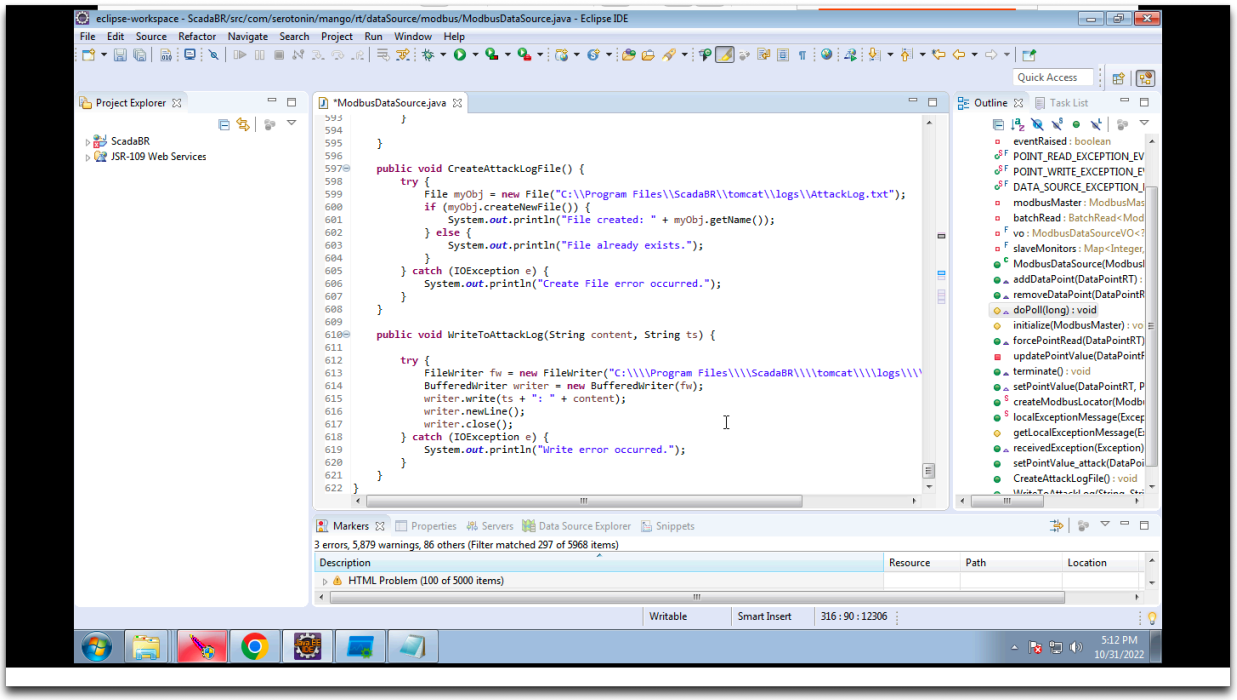
I wrote some code to read PLC data after ScadaBR did TCP Send (READ)



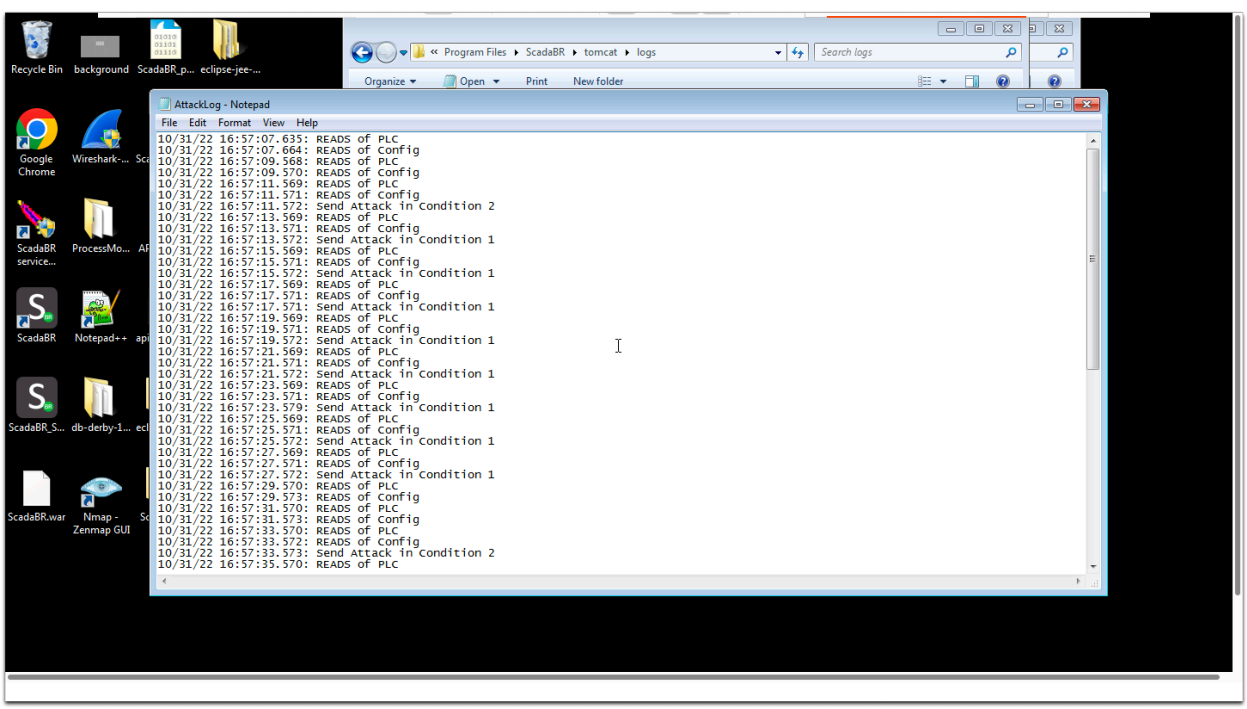
Here's the code to read memory config and decide if it's condition 1 attack or condition 2 attack.



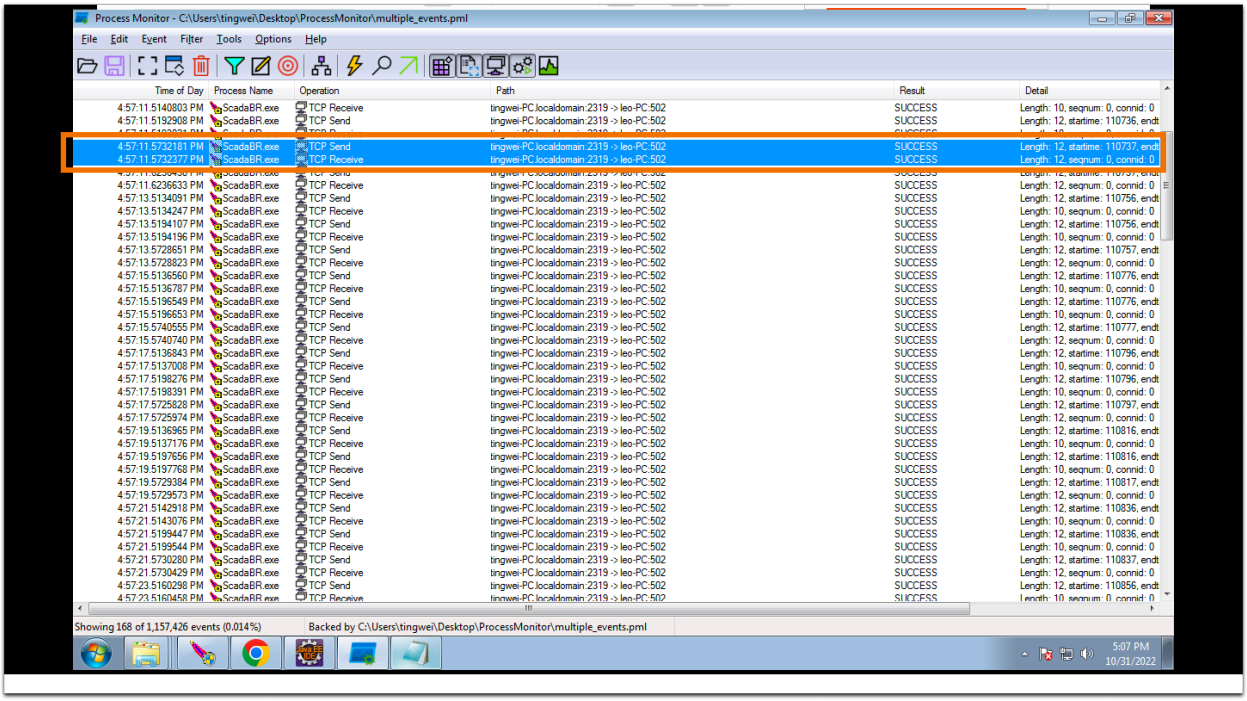
Here's the code to create the Attack Log file and write into it.

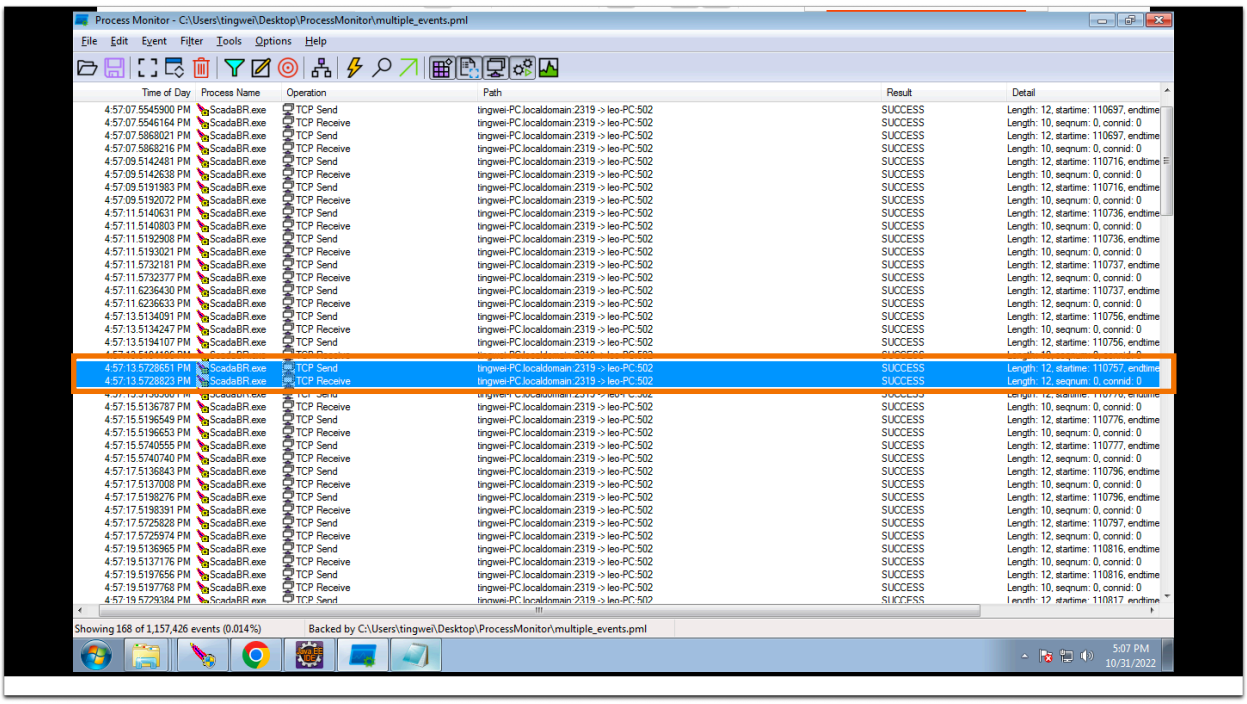


Here's the Attack Log. There are timestamps and corresponding actions.



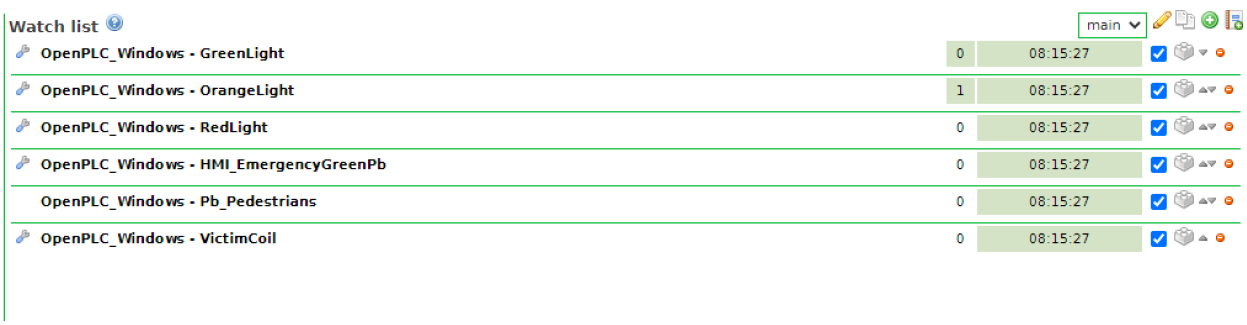
It aligns with the attack timestamp observed from Process Monitor.





On ScadaBR, we can see that the value of VictimCoil indeed change after the attack.

Before attack



After attack