CTF 1

**PLC 1 Tasks**

We ran the generated\_plc1.st in the openplc runtime and it was working fine until we generated the attack. In generated\_plc1.st code, we find that this treatmentComplete variable is not becoming true at all. It is initialized at false and never changed within the code. So, we set up the range and the plc will keep running until it exceeds the range. You will find the code inside the plc1 folder namely `generated\_plc1.st`

PLC 1 attack:

As we know the plc will keep running until the range reaches 100. So, we wrote a script that will stop the plc at the beginning. The malicious code is present inside plc1 folder `malicious-attack1.st`

Again, we can change the range sensor value to generate the attack by subtracting 1 from it. After this, we can disturb the plc movement while it’s running. The malicious code is present inside pcl folder with name `malicious-attack2.st`.

PLC 1 defense:

The attack can’t happen within the plc until it matches certain conditions. So, we made some strict conditions which should be met and the plc will behave accordingly. The defense code is present inside the plc folder with name `generated\_plc1-defenses.st`.

**PLC 2 Tasks**

We ran the generated\_plc2.st file in the openplc runtime and it is working just fine. So, we focused on generating the attacks and defending against those attacks.

PLC 2 attack:

In the plc2, we figured that the

PLC 2 attack (PLC2RedDoserOff.py):

In the plc2 code, I can see that the red doser value is stored at register 6. So, we turned off the red doser in the middle of the system. Which is an unexpected behaviour.

Also, we turned it on with PLC2RedDoserOn.py when it was supposed to be turned off.

Again, with the PLC2TurnOnPump.py file, we turned on the pump when it should be at stop condition.

PLC 2 Defenses: