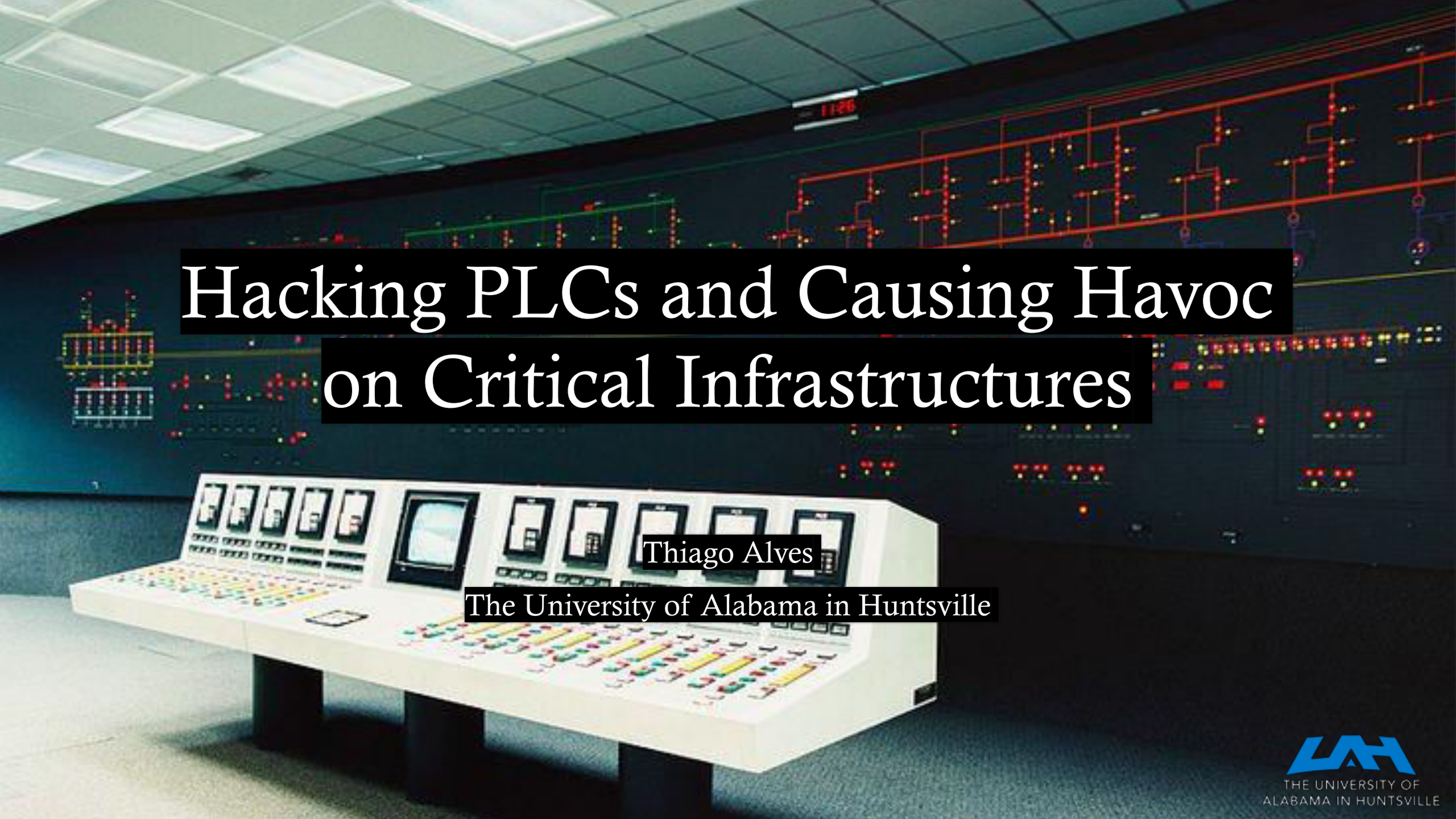


Hacking PLCs and Causing Havoc on Critical Infrastructures

Thiago Alves

The University of Alabama in Huntsville

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Who am I?

- Ph.D. Student at the University of Alabama in Huntsville
- SCADA Cyber Security Researcher at the Center for Cybersecurity Research and Education (CCRE)
- Creator of the OpenPLC Project: www.openplcproject.com – The first fully open source Programmable Logic Controller

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Background Info
- Attack Vectors
- Demos

What Is a PLC?

Programmable Logic Controller

- **Digital Computer** used on Automation
- Input modules **read data** from sensors
- User program **decides what to do** based on the input data
- Output modules **control actuators** on the physical system

Programmable Logic Controller

Programmable Logic Controller

Problems with current PLCs

- **Expensive!**
- Rely on legacy technology
- Use unsecure protocols to communicate
- Most companies rather patch than redesign
- Closed source

OpenPLC – An Open Source Controller

http://www.openplcproject.com

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

OpenPLC – Main Components

Runtime

Editor

GUI Builder

OpenPLC Runtime

Supported platforms

Raspberry Pi

UniPi

PiXtend

ESP8266

Arduino

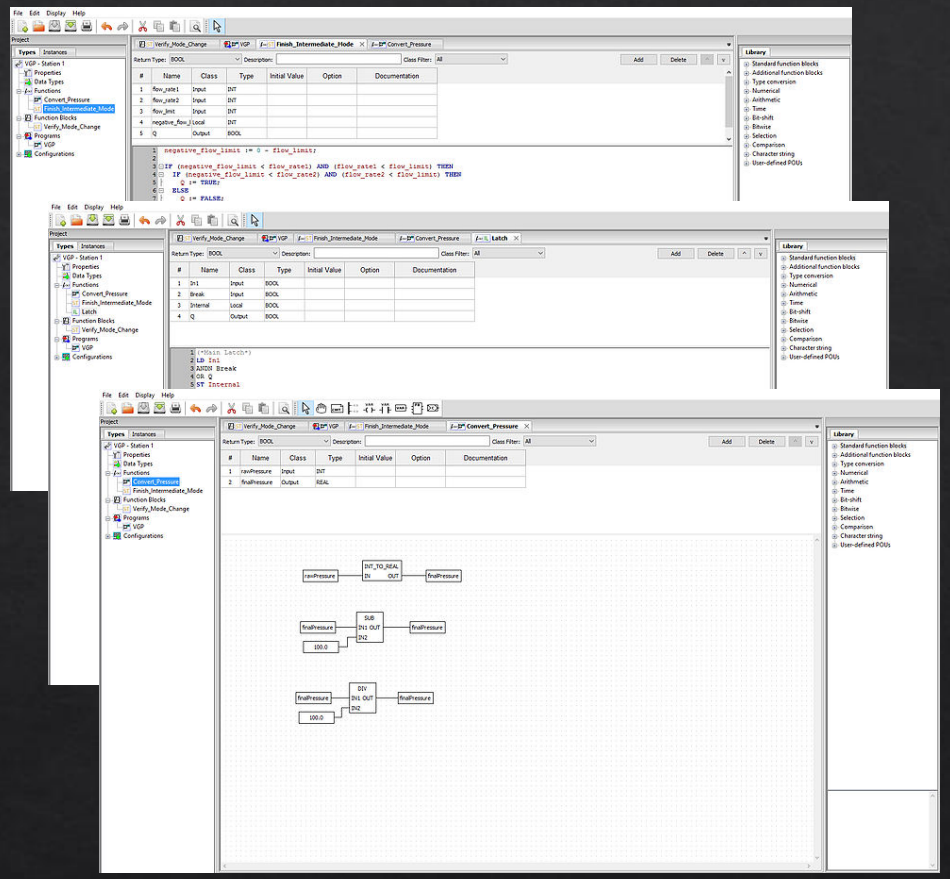
FreeWave Zumlink

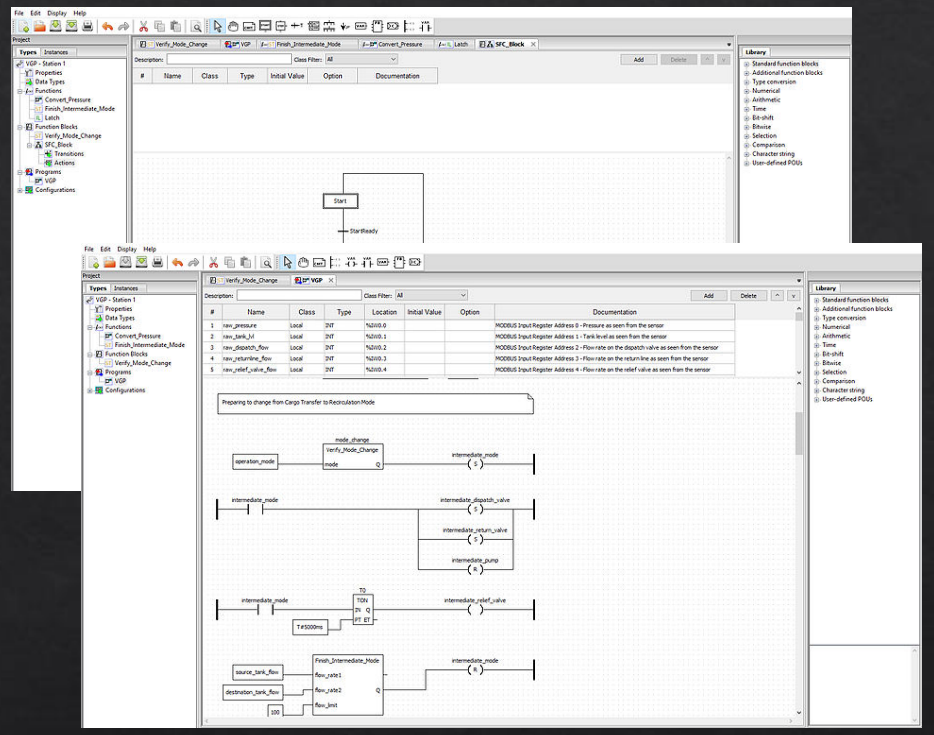
Windows (soft-PLC)

Linux (soft-PLC)

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

OpenPLC Editor

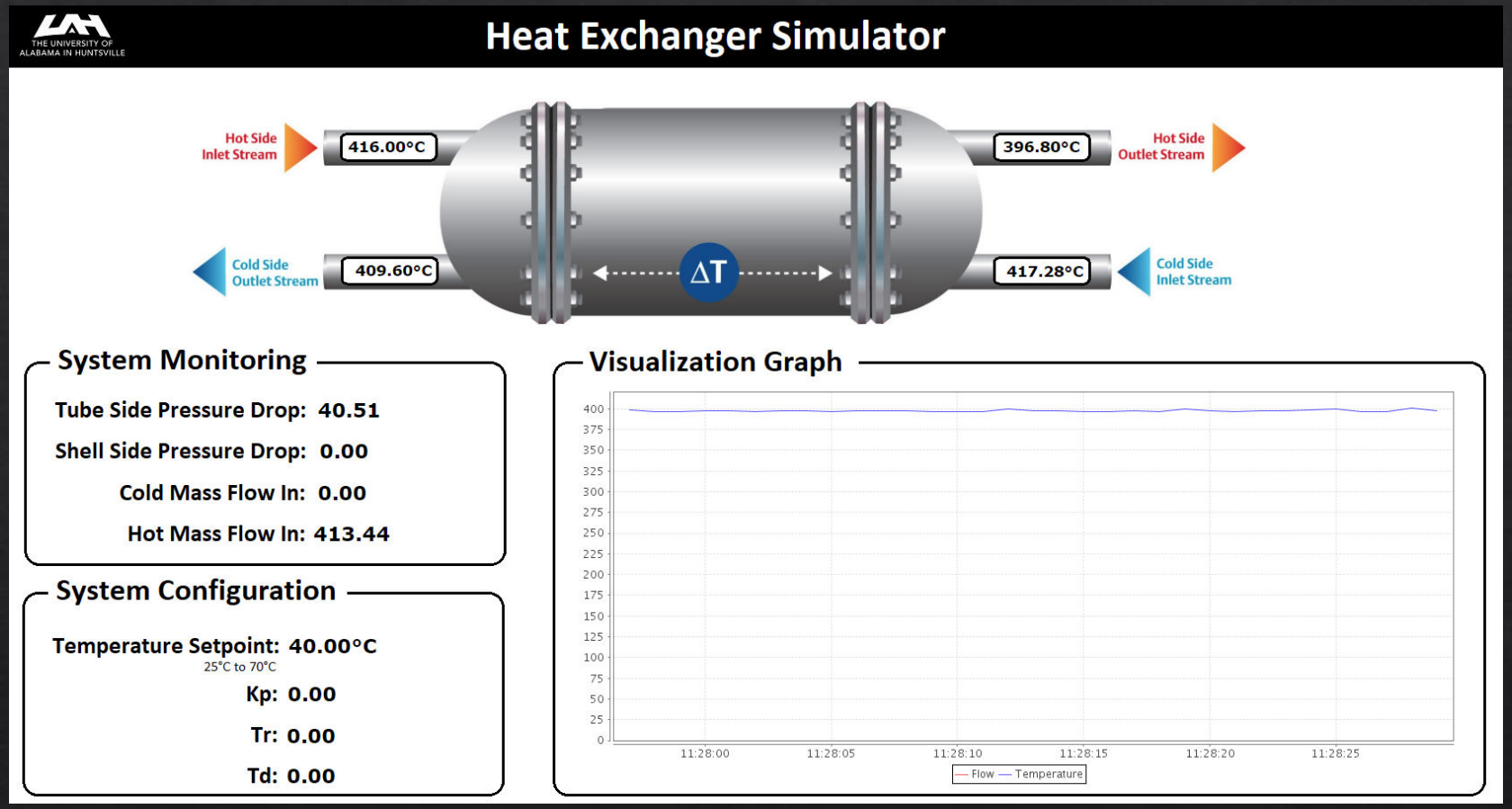




Supports all five IEC 61131-3 programming languages

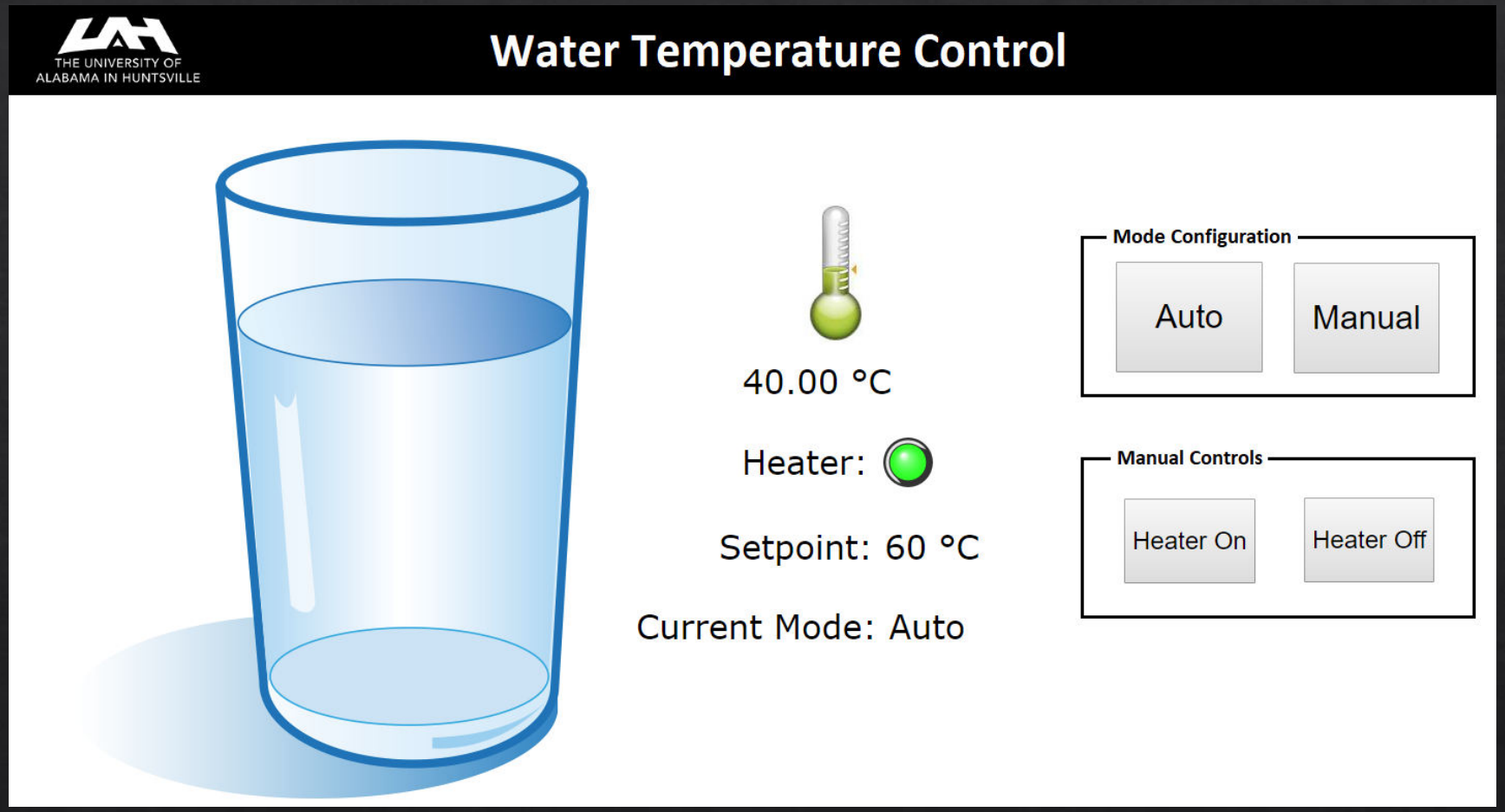
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

ScadaBR



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

ScadaBR


THE UNIVERSITY OF ALABAMA IN HUNTSVILLE
Water Temperature Control
40.00 °C
Heater:
Setpoint: 60 °C
Current Mode: Auto
Mode Configuration
Auto
Manual
Manual Controls
Heater On
Heater Off

PLC Communications: SCADA Protocols

Problems with current SCADA Protocols

- Most protocols are derived from legacy Serial RS-485 networks
- No authentication
- No integrity (prevent tampering)
- No confidentiality

Most Popular SCADA Protocol: Modbus

- Very simple
- Based on serial networks
- Commands encoded into Function Codes
- Open: no licensing required

Modbus Frame

Slave ID: Unique address from 1 to 247

Function Code: Tells the slave which command to execute

Data: Self-explanatory!

CRC: Cyclic Redundancy Check used for error detection

Most used Function Codes

FC01: Read digital outputs

FC02: Read digital inputs

FC03: Read analog outputs

FC04: Read analog inputs

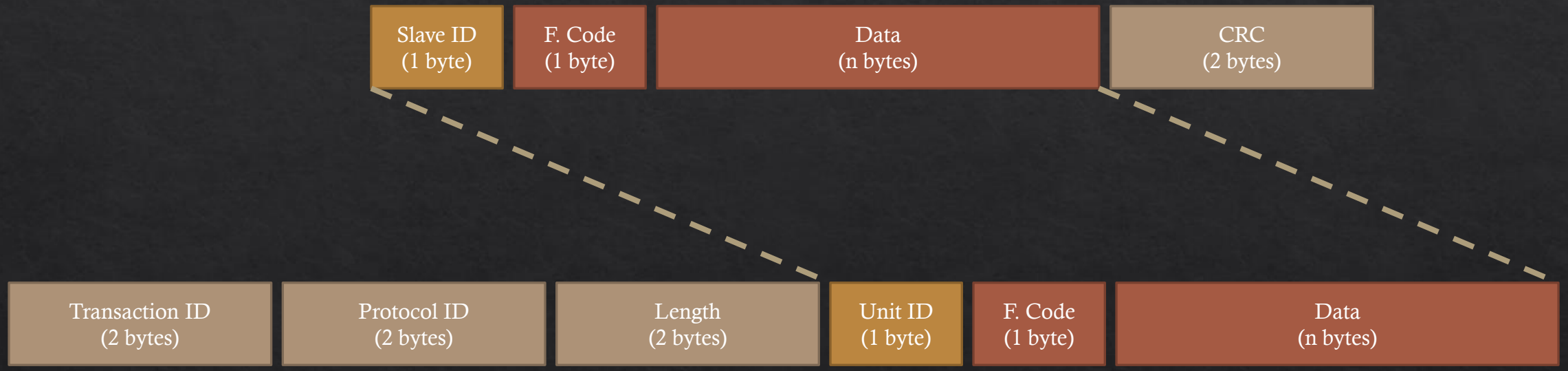
FC05: Write single digital output

FC06: Write single analog output (or register)

FC15: Write multiple digital outputs

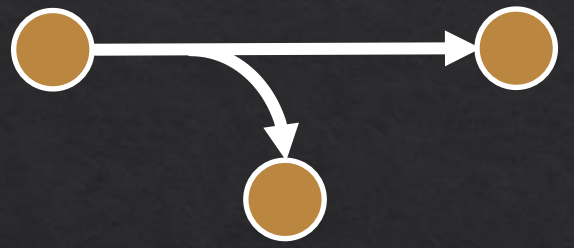
FC16: Write multiple analog outputs (or registers)

Modbus Frame on TCP

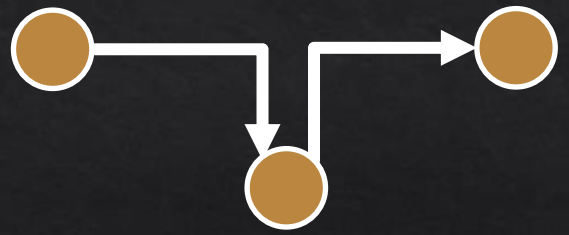


Attack Scenarios

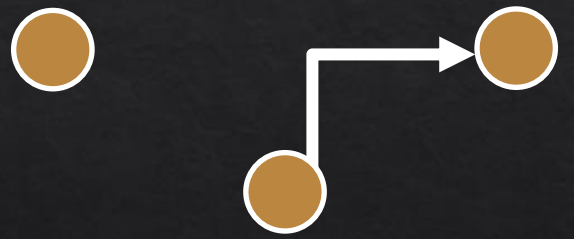
Interruption



Interception



Modification



Injection

Demo: Live Injection Attack

One more thing…

MICROLOGIX DEADLY PACKETS

Transaction ID	Protocol ID	**Length**	Unit ID	Func. Code	Coil Address	Status
00ad	0000	00ff	01	05	0043	ff00
071b	0000	0006	01			

Demo: Remotely Crashing a Micrologix 1400

And one last thing…

Schneider Unity Protocol

- Runs on top of Modbus/TCP
- Uses Function Code 0x5A
- Used to **configure** and **monitor** the PLC
- Proprietary and undocumented protocol (security by obscurity)

Unity's Function Codes

0x01 - INIT_COMM: Initialize a UMAS communication
0x02 - READ_ID: Request a PLC ID
0x03 - READ_PROJECT_INFO: Read Project Information
0x04 - READ_PLC_INFO: Get internal PLC Info
0x06 - READ_CARD_INFO: Get internal PLC SD-Card Info
0x0A - REPEAT: Sends back data sent to the PLC (used for sync)
0x10 - TAKE_PLC_RESERVATION: Reserve the PLC to a station
0x11 - RELEASE_PLC_RESERVATION: Release the reservation
0x12 - KEEP_ALIVE: Keep alive message
0x20 - READ_MEMORY_BLOCK: Read a memory block from PLC
0x22 - READ_VARIABLES: Read System bits, Words and variables
0x23 - WRITE_VARIABLES: Write System bits, Words and variables
0x24 - READ_COILS_REGISTERS: Read coils and holding registers
0x25 - WRITE_COILS_REGISTERS: Write coils and holding registers

0x30 - INITIALIZE_UPLOAD: Initialize upload
0x31 - UPLOAD_BLOCK: Upload a block to the PLC
0x32 - END_STRATEGY_UPLOAD: Finish Upload
0x33 - INITIALIZE_DOWNLOAD: Initialize download
0x34 - DOWNLOAD_BLOCK: Download a block
0x35 - END_STRATEGY_DOWNLOAD: Finish download
0x39 - READ_ETH_MASTER_DATA: Read Ethernet Master Data
0x40 - START_PLC: Starts the PLC
0x41 - STOP_PLC: Stops the PLC
0x50 - MONITOR_PLC: Monitors variables, Systems bits and words
0x58 - CHECK_PLC: Check PLC Connection status
0x70 - READ_IO_OBJECT: Read IO Object
0x71 - WRITE_IO_OBJECT: Write IO Object
0x73 - GET_STATUS_MODULE: Get Status Module

Source: lirasenlared.xyz

Unity's Function Codes

0x01 - INIT_COMM: Initialize a UMAS communication

0x02 - READ_ID: Request a PLC ID

0x03 - READ_PROJECT_INFO: Read Project Information

0x04 - READ_PLC_INFO: Get internal PLC Info

0x06 - READ_CARD_INFO: Get internal PLC SD-Card Info

0x0A - REPEAT: Sends back data sent to the PLC (used for sync)

0x10 - TAKE_PLC_RESERVATION: Reserve the PLC to a station

0x11 - RELEASE_PLC_RESERVATION: Release the reservation

0x12 - KEEP_ALIVE: Keep alive message

0x20 - READ_MEMORY_BLOCK: Read a memory block from PLC

0x22 - READ_VARIABLES: Read System bits, Words and variables

0x23 - WRITE_VARIABLES: Write System bits, Words and variables

0x24 - READ_COILS_REGISTERS: Read coils and holding registers

0x25 - WRITE_COILS_REGISTERS: Write coils and holding registers

0x30 - INITIALIZE_UPLOAD: Initialize upload

0x31 - UPLOAD_BLOCK: Upload a block to the PLC

0x32 - END_STRATEGY_UPLOAD: Finish Upload

0x33 - INITIALIZE_DOWNLOAD: Initialize download

0x34 - DOWNLOAD_BLOCK: Download a block

0x35 - END_STRATEGY_DOWNLOAD: Finish download

0x39 - READ_ETH_MASTER_DATA: Read Ethernet Master Data

0x40 - START_PLC: Starts the PLC

0x41 - STOP_PLC: Stops the PLC

0x50 - MONITOR_PLC: Monitors variables, Systems bits and words

0x58 - CHECK_PLC: Check PLC Connection status

0x70 - READ_IO_OBJECT: Read IO Object

0x71 - WRITE_IO_OBJECT: Write IO Object

0x73 - GET_STATUS_MODULE: Get Status Module

Source: lirasenlared.xyz

Demo: Exploiting the Unity Protocol

Thanks!

Thiago Alves
www.openplcproject.com

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE