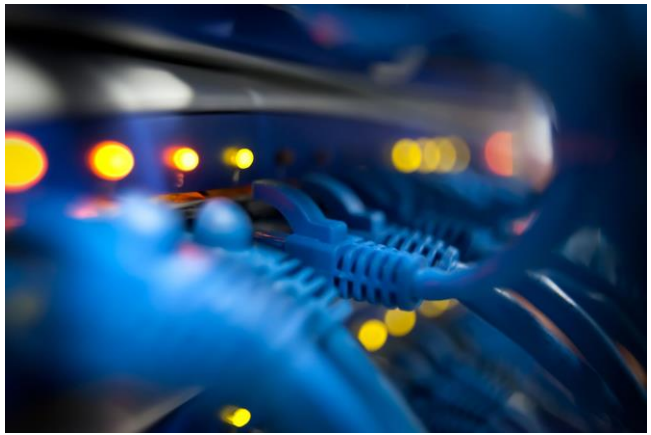
Securing the bridge between the cyber and physical worlds: Cybersecurity for PLCs

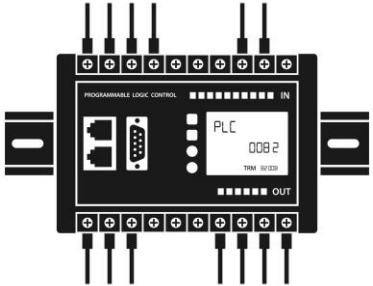
David Formby, PhD
Cofounder and CEO/CTO Fortiphyd Logic

Background

Programmable Logic Controllers (PLCs)

Cyber/IT





Physical/OT

Bridge between cyber and physical

Insecure by design

Most attention is paid to network

FORTIPHYD LOGIC

Outline

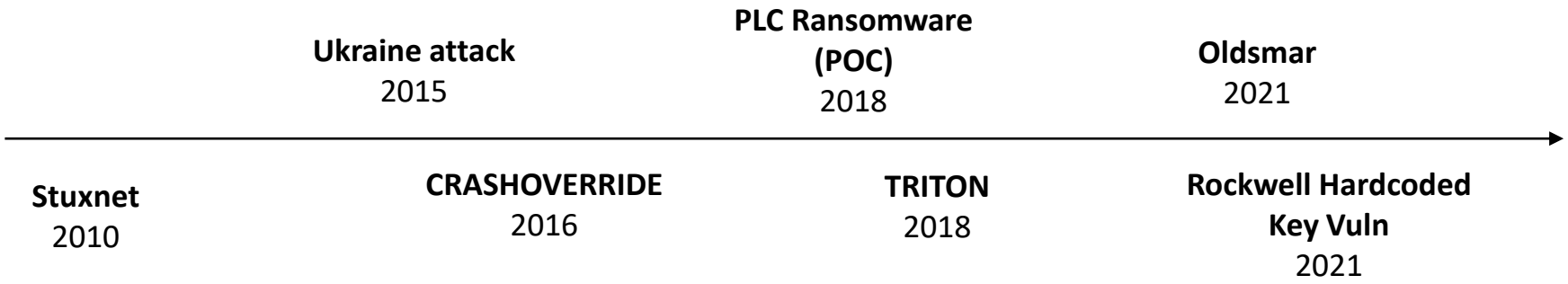
- Bio
- Threats to PLCs
- Secure PLC Coding Practices
- PLC Program Anomaly Detection

Bio – David Formby

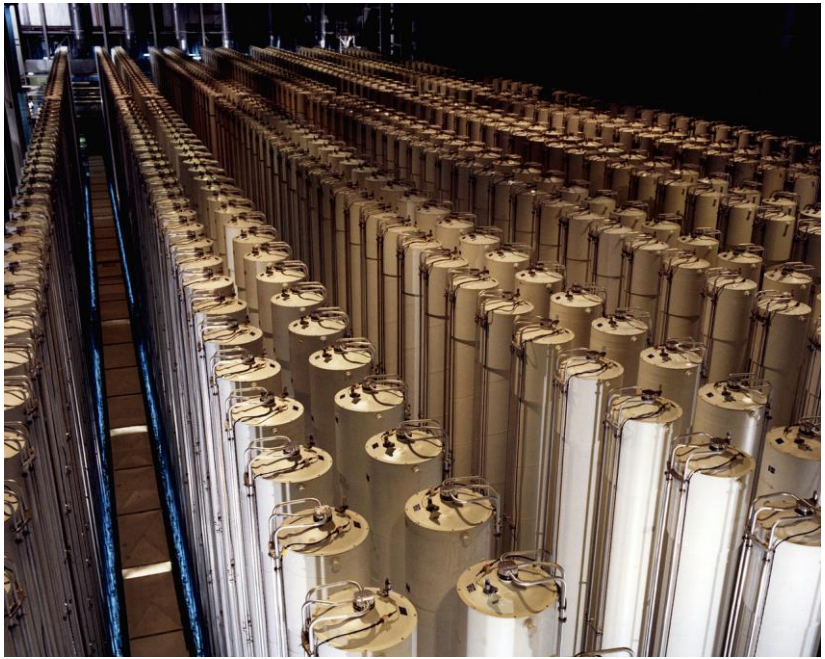
- PhD Electrical & Computer Engineering, Georgia Tech
  - PLC ransomware worm
  - PLC anomaly detection
  - Open source ICS security simulation
- Dozens of ICS-CERT vulnerabilities
- Cofounder and CEO/CTO of Fortiphyd Logic
  - Network monitoring
  - PLC endpoint security
  - ICS security training

Threats to PLCs

Threat Timeline for PLCs (Purdue Level 1)



Stuxnet (2010)



- Target
  - Iranian nuclear program
- Key points
  - Jumped “air-gap”
  - Reprogrammed PLC, and hid the changes
- Effects
  - Destroyed up to 1000 centrifuges

Ukraine Attack (2015)

- Target
  - Ukrainian power distribution companies
- Key points
  - Stole operator remote credentials
  - Used existing HMI
  - Complex amplified attack
    - DoS telephone service
    - Bricked RTUs, disabled UPS
- Effects
  - 225,000 customers lost power

CRASHOVERRIDE (2016)

- Target
  - Ukrainian transmission substation
- Key points
  - Highly modular, complex malware
  - Deep understanding of ICS protocols
- Effects
  - Section of capital city, Kiev, lost power for one hour
  - Large-scale testing of malware not serious attempt

PLC Ransomware (Proof-of-concept)

- Georgia Tech research project
- Lack of ICS attacks because financial motivation not secure
  - Ransomware could be effective monetization of insecurity
- Worm compromised one PLC and spread to others, locking out operators

Trisis/Triton/HatMan (2018)

- Target
  - Saudi Arabian oil facility
- Key points
  - Reverse engineered programming software
  - PLC “rootkit” remains even if switch is in Run Mode
  - Only detected because of attacker programming mistake
- Effect
  - Emergency shutdown and long debug time

Oldsmar

- Target
  - Small water utility in Florida
- Key points
  - Remote access to HMI
  - Credentials compromised
  - Attacker just played around on HMI
- Effect
  - Changed setpoint to dangerous levels, but operator immediately corrected

Rockwell Hardcoded Key Vulnerability

- Vulnerable PLCs
  - Entire Logix line of PLCs
- Key Points
  - Hard coded key vulnerability
  - Anyone on network with key can reprogram PLC
  - Official response – no patch in sight
- Effect
  - PLCs are still insecure by design

Top 20 Secure PLC Coding Practices

FORTIPHYD LOGIC

Motivation

- Computer programming has well established guidelines for “secure coding” to mitigate vulnerabilities and produce reliable software
  - Software development life cycle (SDLC)
  - OWASP Secure Coding Practices (web applications)
  - Secure Coding Guidelines for Java SE
  - Microsoft Security Development Lifecycle
- PLC programming lacks similar standards

Secure PLC Coding Practices

- Presentation at S4 Conference by Jake Brodsky
  - Basic tips on what PLC programmers can do to add security
- First attempt at formalizing the tips
  - Organizers
    - Admeritia (Vivek Ponnada and Sarah Fluchs)
    - S4 Conference (Dale Peterson)
    - Other volunteers
  - June 2021
  - Top 20 Secure PLC Coding Practices

License

Security Objectives

- **Integrity** (Practices 1-12)
  - PLC Logic – Detecting/preventing unauthorized changes to program
  - PLC variables - Detecting/preventing unauthorized changes to variables
  - IO values - Detecting/preventing unauthorized changes to IO values
- **Hardening** (Practices 13,14)
  - Make initial compromise more difficult
- **Resilience** (Practices 15,7)
  - Assuming compromise, mitigate threat to continuing mission/operation
- **Monitoring** (Practices 16-20)
  - Detect incident as soon as possible to mitigate damage

Benefits

- Security
  - Mitigating threat from intentional attackers
- Reliability
  - Mitigating threat from accidents
- Maintenance
  - Making code easier to debug and maintain

Additional References

- MITRE ATT&CK for ICS
  - Formal descriptions of attacks on ICS and mitigations
    - Tactics, Techniques, Procedures, Mitigations
- ISA 62443
  - 3-3 System security requirements and security levels
  - 4-1 Secure product development lifecycle requirements
  - 4-2 Technical security requirements for IACS components
- MITRE CWE
  - Common weakness enumeration

Limitations

- Secure PLC Coding Practices are only one layer of defense
  - Good network security and monitoring still necessary
- Most PLCs are still insecure by design
- Secure practices only make it harder for the attacker, not impossible
  - Many can be worked around by attacker with complete control of PLC
  - Monitoring is only useful if someone watches it, or gets alerts
- Only a first pass, still looking for user feedback and refinement

1. Modularize PLC Code

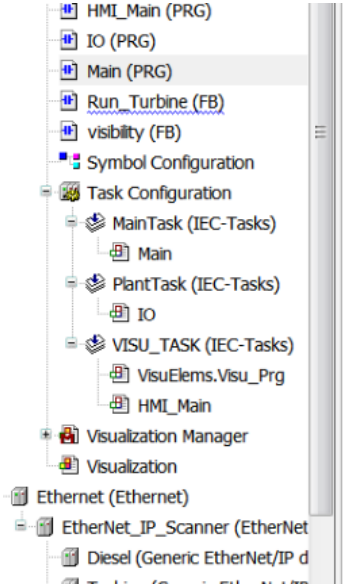
- Break PLC logic down into function blocks or subroutines that are:
  - Re-usable
  - Testable
  - Independent
- For each FB or subroutine:
  - Test thoroughly, ranges of inputs
  - Record execution time, size in rungs or kB, checksum/signature if available
  - Lock with password/certificate if possible and it makes sense
  - Track changes

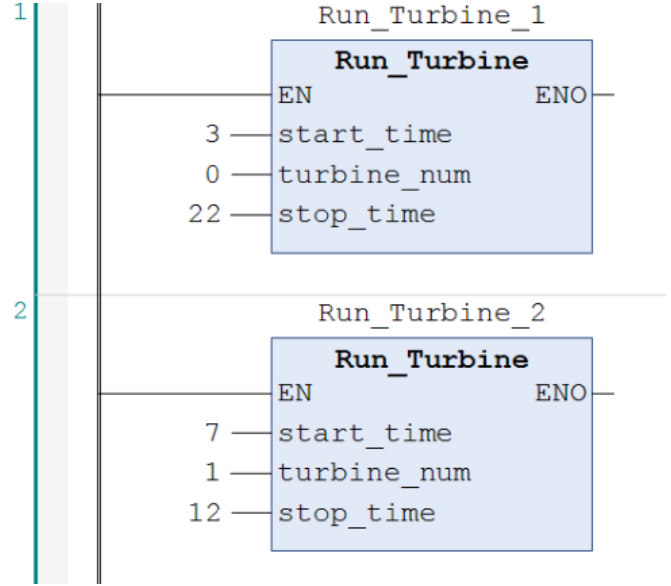
Integrity of PLC Logic

FORTIPHYD LOGIC

1. Modularize PLC Code - Example

- Gas turbine startup sequence
  - Function block for general turbine, instantiate for each turbine
    - Made of smaller function blocks for each component





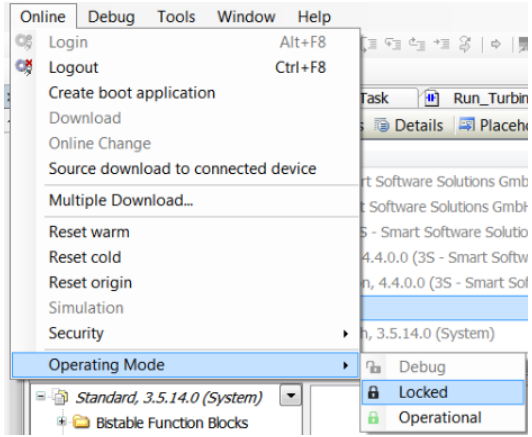
2. Track operating modes

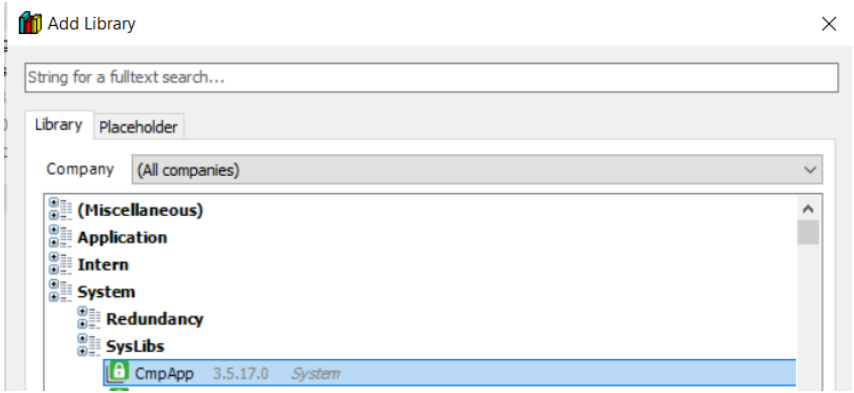
- Most PLCs have RUN and PROGRAM modes, and only can be programmed with normal software in PROGRAM mode
- Keep PLCs in RUN mode as much as possible
- Display current mode on HMI
- Alarm on mode changes, and continuously for PROGRAM mode
  - Exception – extended maintenance windows, extra precautions elsewhere

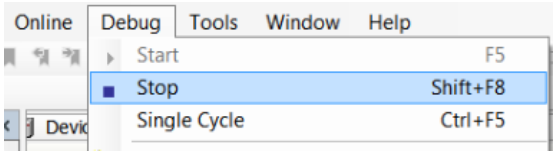
Integrity of PLC Logic

FORTIPHYD LOGIC

2. Track operating modes - Example





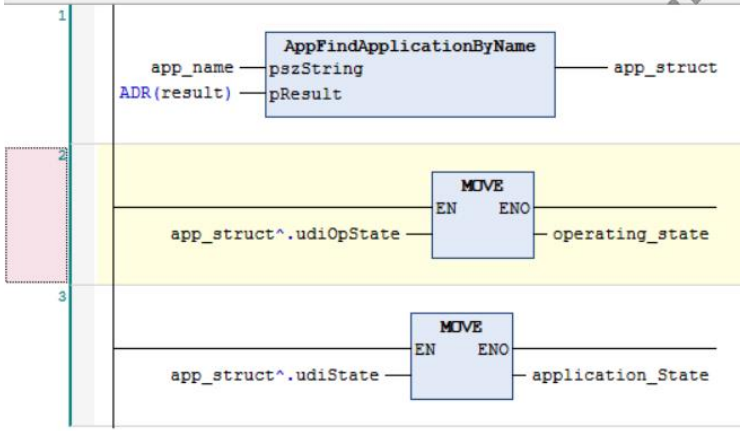


FORTIPHYD LOGIC

2. Track operating modes - Example

PLC_PRG | Library Manager

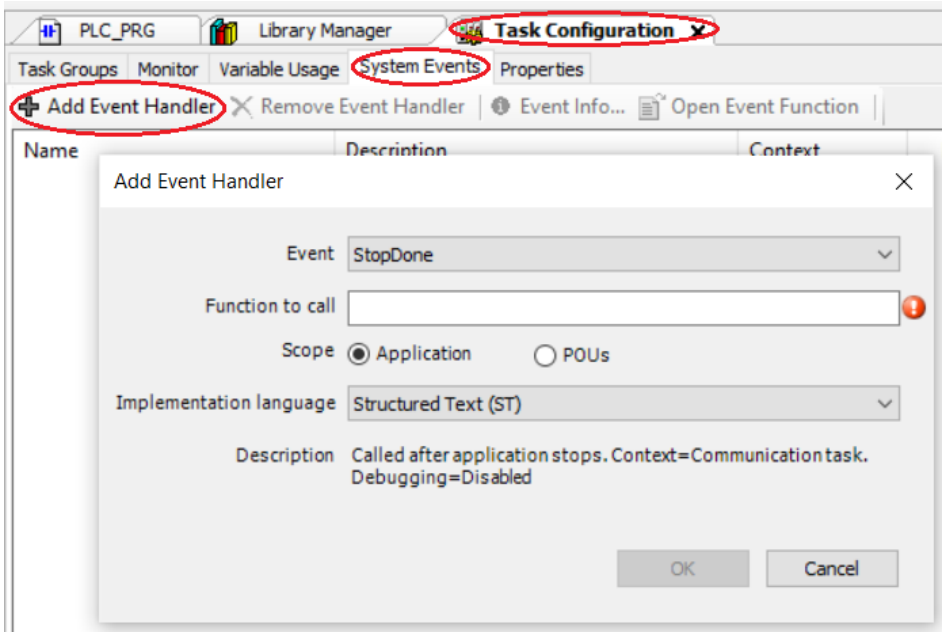
```
1  PROGRAM PLC_PRG
2  VAR
3      operating_state: UDINT;
4      application_State: UDINT;
5      app_name :STRING := 'MyApp';
6      result : RTS_IEC_RESULT;
7      app_struct : POINTER TO Application;
8  END_VAR
```



Name	Type	Inherited from	Address	Initial	Comment
OS_NONE	UDINT			16#0	
OS_PROGRAM_LOADED	UDINT			16#1	
OS_DOWNLOAD	UDINT			16#2	
OS_ONLINE_CHANGE	UDINT			16#4	
OS_STORE_BOOTPROJECT	UDINT			16#8	
OS_FORCE_ACTIVE	UDINT			16#10	
OS_EXCEPTION	UDINT			16#20	
OS_RUN_AFTER_DOWNLOAD	UDINT			16#40	
OS_STORE_BOOTPROJECT_ONLY	UDINT			16#80	

Name	Type	Inherited from	Address	Initial	Comment
AS_NONE	UDINT			16#0	Application states
AS_RUN	UDINT			16#1	
AS_STOP	UDINT			16#2	
AS_DEBUG_HALT_ON_BP	UDINT			16#3	
AS_DEBUG_STEP	UDINT			16#4	
AS_SINGLE_CYCLE	UDINT			16#5	

2. Track operating modes - Example



3. Leave operational logic in the PLC

- HMIs have some programmability
- Leave safety and key operational logic in PLC
  - Summarizing/totaling logic
  - Logic to enable/disable buttons (timers, counters…)
  - Thresholds to trigger alarms

Integrity of PLC Logic

5. Use crypto/checksum integrity checks

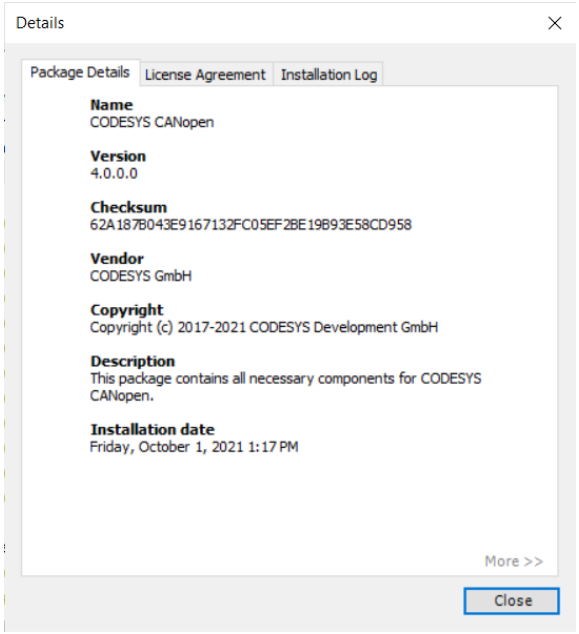
- Cryptographic hashes, checksums, audit values, signatures
  - Methods for “fingerprinting” PLC program and configuration
  - Available on most PLCs, some more secure than others
- Log checksum/hash as part of SAT, documentation of final product
  - Alarm when it changes

Integrity of PLC Logic

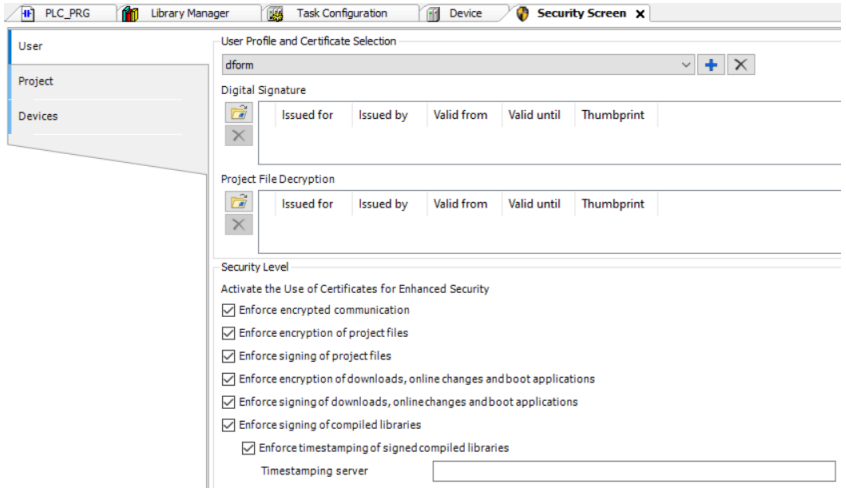
FORTIPHYD LOGIC

5. Use crypto/checksum integrity checks

Package Checksums



Signing Project Files with Certificates



https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_encrypting_signing_with_cerificates/

FORTIPHYD LOGIC

7 / 11 / 12 - Plausibility

- Paired IO – physically cannot happen at the same time
  - Start/stop, forward/reverse, open/close
- Cross check multiple sources of sensor data for related plausibility
  - Compare integrated/summed measurements with instantaneous values
    - Flow rates with volumes
  - Compare different sensors that should be related
    - Valve open & flow rate > 0 / valve closed & flow rate = 0
- Only allow inputs that are physically possible/safe
  - Set a timer for longer operations to verify it completes in physically plausible amount of time

Integrity of PLC Variables
Resilience

FORTIPHYD LOGIC

7. Validate and alert for paired IO

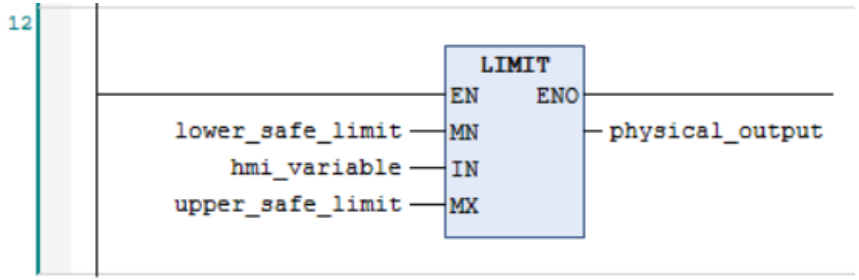
8. Validate HMI inputs at PLC

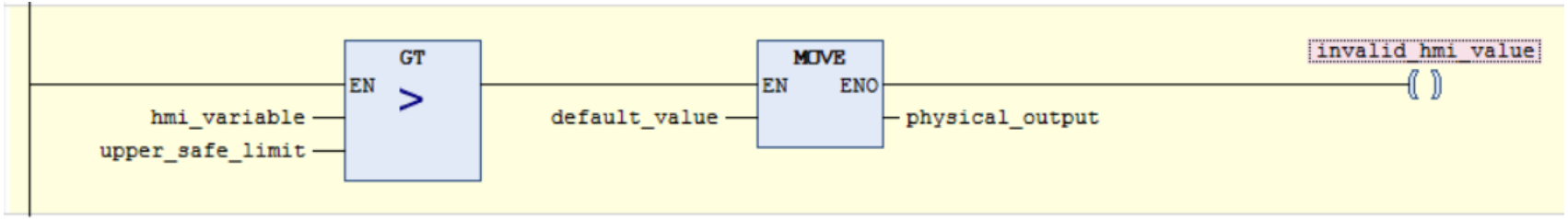
- Values should be limited to safe ranges **not just** at HMI, but also in the PLC logic
- Limiting ranges in the HMI only prevents accidental attacks
  - Moderately skilled attackers can bypass HMI checks
- If invalid value is received in PLC, log it
  - Use a default safe value
  - Use last valid value
  - Limit to closest max

Integrity of PLC Variables

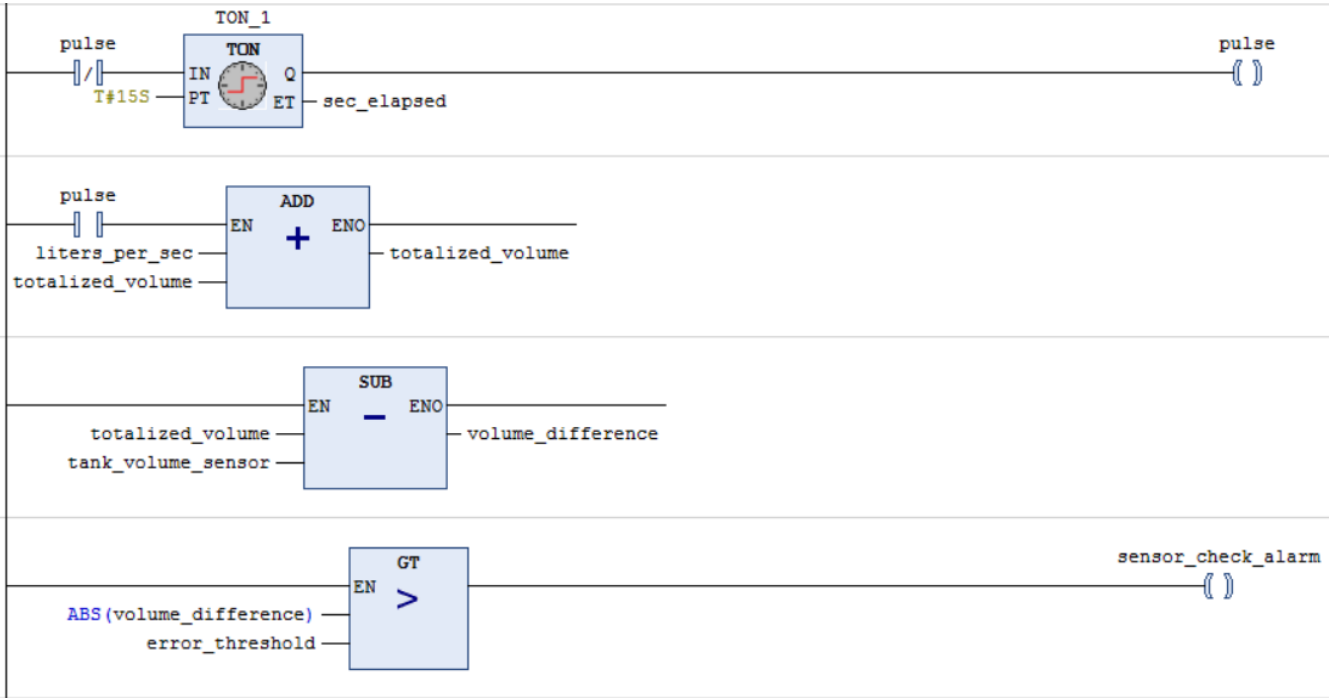
FORTIPHYD LOGIC

8. Validate HMI inputs at PLC

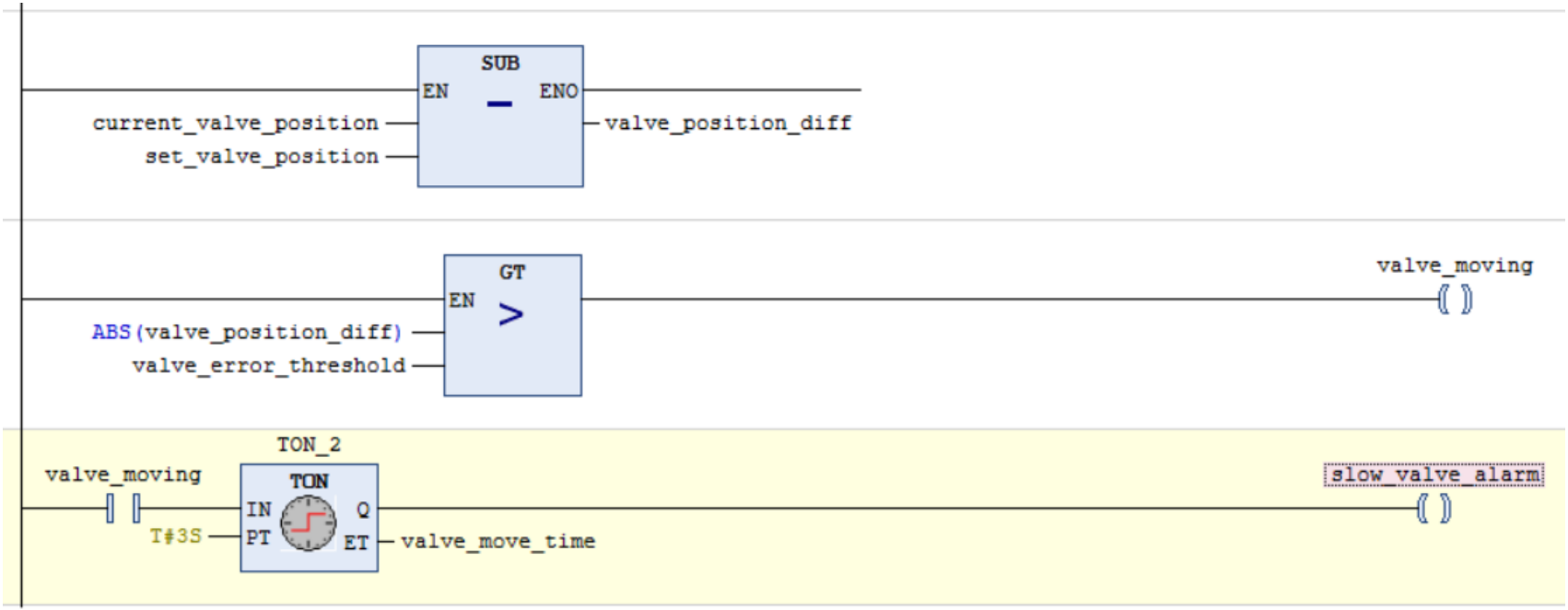




11. Instrument for plausibility checks



12. Validate inputs for physical plausibility



13. Disable unneeded ports and protocols

- PLCs, especially newer ones, come with multiple protocols supported
- Disable ports and protocols that are not needed
  - Web server configuration
  - SNMP, Telnet, FTP
  - Modbus, OPC UA
- Develop data flow diagram showing required communication for PLCs and physical ports

Hardening

FORTIPHYD LOGIC

13. Disable unneeded ports and protocols - Examples

- Disable OPC UA if not used
- Don’t add protocols unless you are using them
- Configure host firewall to only allow needed ports for CODESYS and whatever protocols

15. Define a safe process state for reboots

- PLCs can unexpectedly reboot
  - Accidental power loss
  - Exploit attempt
- In case of PLC reboot, start into a safe output state
  - Valves closed/open, motors on/off

Resilience

FORTIPHYD LOGIC

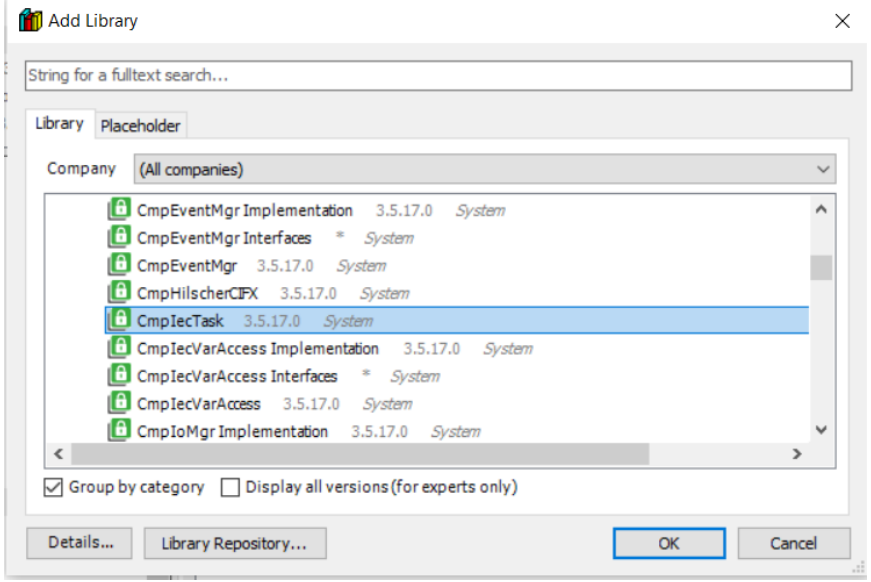
16. Summarize cycle times and trend them

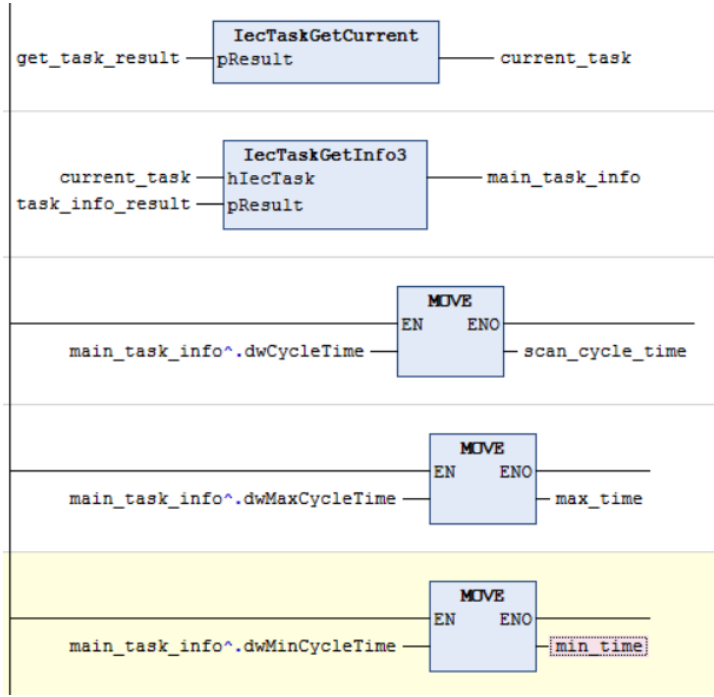
- Program execution times usually measured as system variable
- Program execution times are relatively consistent unless
  - PLC program changes
  - Physical process changes significantly
  - Network changes significantly
- Monitor average, max, and min scan cycle times
  - Trend on HMI to visually detect unusual program behavior
- Automatically detect changes
  - Fortiphyd's **LogicWatch Pro**

Monitoring

FORTIPHYD LOGIC

16. Summarize cycle times and trend them





FORTIPHYD LOGIC

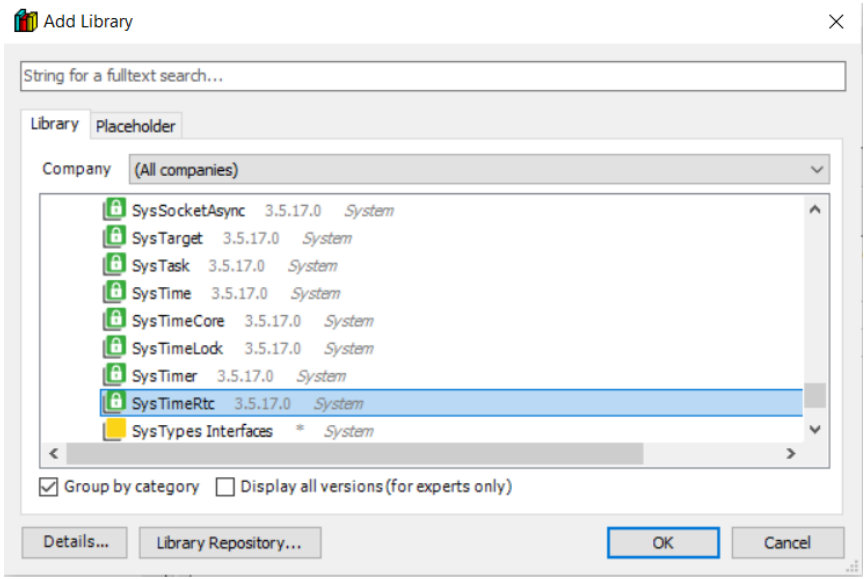
17. Log uptime and trend it

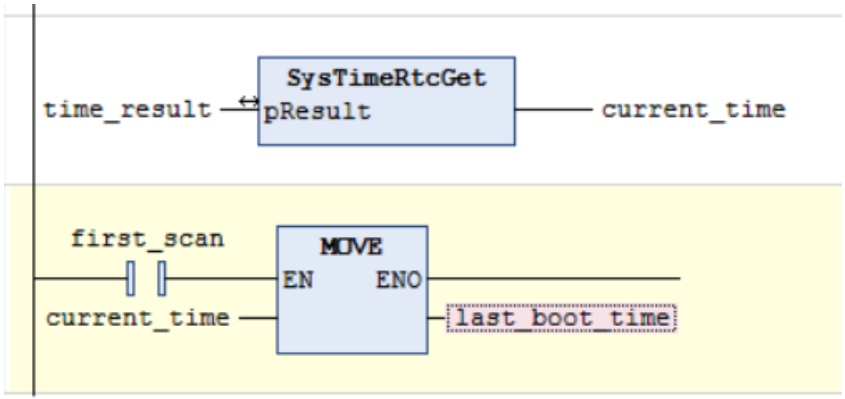
- PLC reboots can indicate maintenance problems or exploit attempts
- Monitor time since last reboot to detect reboots
  - In PLC logic
  - From network using SNMP

Monitoring

FORTIPHYD LOGIC

17. Log uptime and trend it





FORTIPHYD LOGIC

18. Log hard stops and trend them

- Hard stops from faults can indicate maintenance problems or attacks
- Log hard stops for investigation before continuing operations
  - Accurate timestamping important for debugging
  - Log which error codes

Monitoring

FORTIPHYD LOGIC

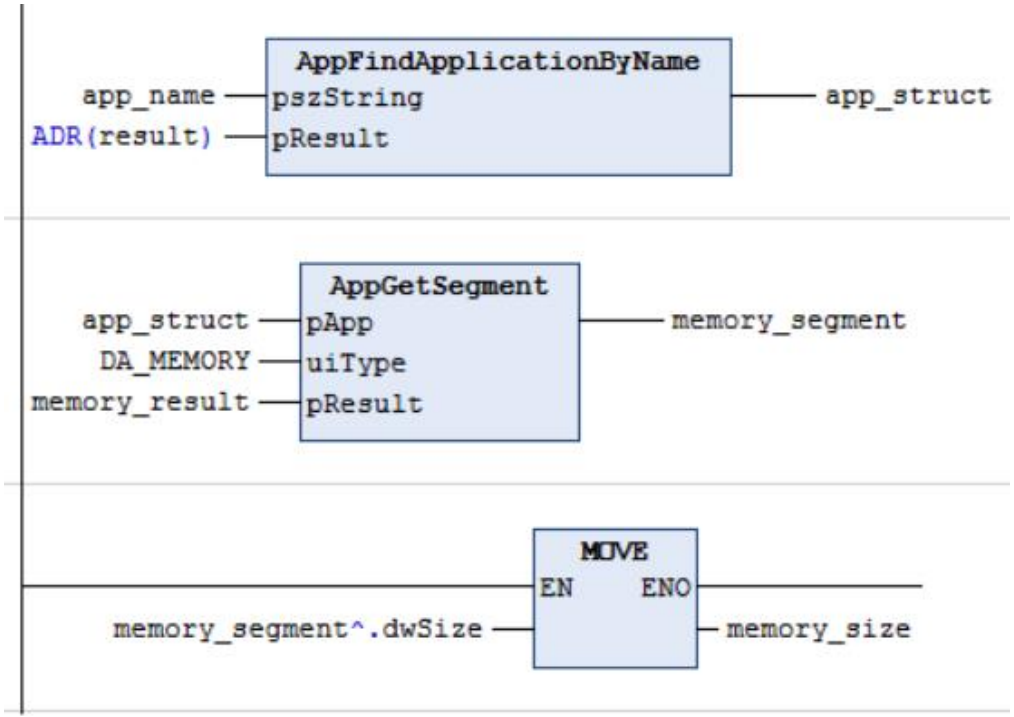
19. Monitor memory usage and trend it

- Two types of memory
  - Storage
  - RAM
- Change in program memory indicates change in program
  - More or less running code
- Track what is normal and alert on thresholds

Monitoring

FORTIPHYD LOGIC

19. Monitor memory usage and trend it



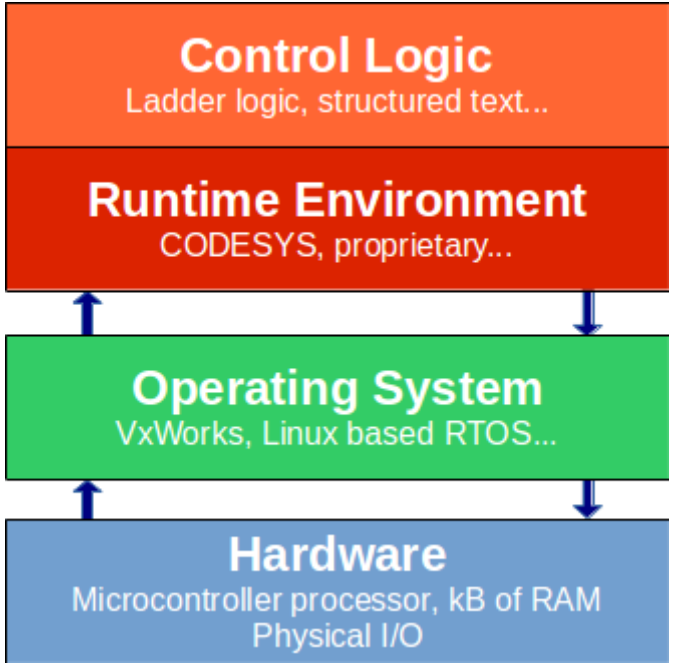
PLC Anomaly Detection

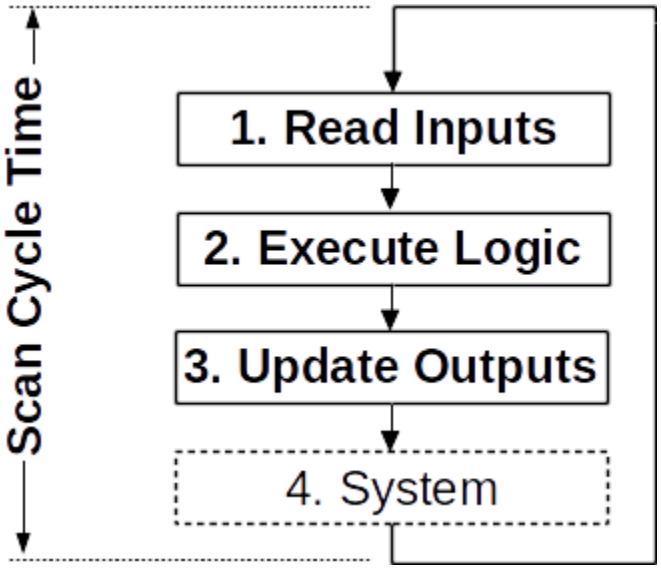
FORTIPHYD LOGIC

Motivation

- Vendors cannot secure their PLCs on their own
- Give operators a tool to add strong security themselves
- Leverage the limitations of PLCs as a strength
  - Single purpose programming -> predictable program execution
  - Slow processors -> changes in programming more noticeable
  - Limited memory -> incapable of storing large amounts of data
- Result
  - “Temporal Execution Behavior for Host Anomaly Detection in Programmable Logic Controllers” IEEE TIFS 2019
  - Patent, *LogicWatch Pro*

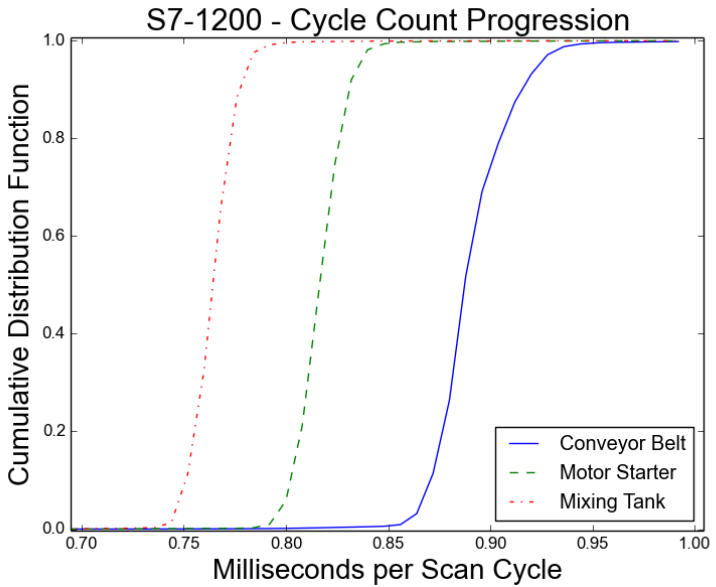
Change Detection





Change Detection

- Change in PLC program results in change in execution time
- Small change not noticeable in one scan cycle
  - Over thousands of cycles, accumulates into detectable change
- Tested across 3 main vendors
  - Rockwell, Siemens, Modicon
- Single instruction change was detectable on sample programs

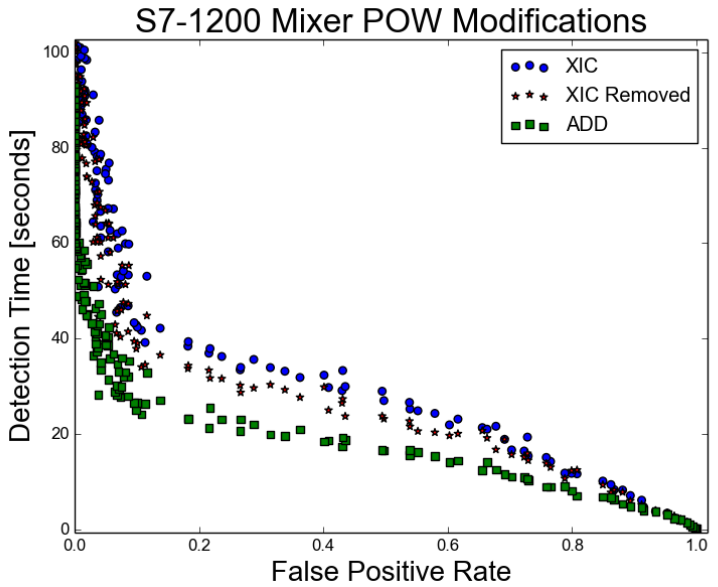


Proof of Work

- Limitations
  - Attacker with knowledge of algorithm can bypass it
  - Harder to fake than static value, but still feasible to repeat previous values
- Proof of work (POW) function
  - Computationally expensive to solve, but easy to verify
  - Give PLC an “alibi”, to prove it was busy doing POW and did not have time to execute anything else

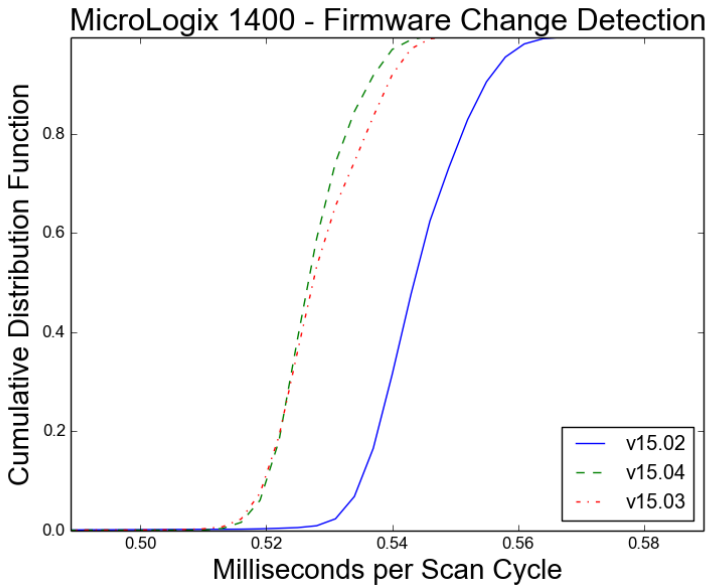
Proof of Work

- Discrete logarithm
  - $x^y \bmod p = z$
  - To solve:
    - General approach – brute force multiply base over and over again
      - O(n), n size of modulus group
    - Best approaches O(sqrt(n))
  - To verify:
    - Exponentiation by squaring - O(log(n))



Firmware Modification

- Changes to underlying firmware contribute to program execution time
- Different firmware versions were found to be detectable changes



White Box Modeling

- Rare branches of code not executed very often
- Some PLCs list instruction execution time
  - With full “white box” knowledge of PLC program, can estimate execution time of rare branches
- Results
  - White box not as accurate, but viable approach

Conclusion

- Threats to PLCs are increasing, but they are still insecure by design
- PLC programmers can add security
  - PLC programming
    - Integrity – Don't trust anything from the network
    - Hardening – Remove unnecessary functionality
    - Resilience – Assume attack/misuse, try to fail safe
    - Monitoring – Trend KPI of PLC to detect incidents
  - PLC program anomaly detection
    - Program execution time and proof of work functions

David Formby
dformby@fortiphyd.com

Twitter: @fortiphyd
LinkedIn

References

- Fortiphyd Training Grounds Course on Secure PLC Coding
  - https://fortiphyd.talentlms.com/catalog/info/id:166
- Top 20 Secure PLC Coding Practices
  - https://plc-security.com/
- ISA 62443 Standards
  - https://www.isa.org/standards-and-publications/isa-standards/find-isa-standards-in-numerical-order
- MITRE ATT&CK for ICS
  - https://collaborate.mitre.org/attackics/index.php/Main_Page
- MITRE CWE
  - https://cwe.mitre.org/