Towards a Highly Secure PLC Architecture for CPS

Nikhil Mahesh,
University of Southern California

Short Presentation
Layered Assurance Workshop (LAW)
Dec 8-9, 2014, New Orleans
Introduction

• Recent attacks on cyber physical system (CPS)
  – Rootkit is planted in PLC, PLCpw
  – Most on power grid
• Discovery of the Stuxnet attack
  – Based on software subversion malware
• Demonstrate vulnerabilities in the current CPS
• Industrial Control Systems (ICS) components
  – Programmable Logic Controller (PLC)
  – Human Machine Interface (HMI)
  – Software development environment.
How to Make CPS Secure?

• Design security in from the beginning
  – System is secure only with respect to a policy
  – Especially include Mandatory Access Control (MAC)
  – Often no explicit security policy in current systems

• Apply proven high assurance technology
  – Reference Monitor security engineering principles

• Create a reusable “Trusted Device”
  – Start with existing proven MAC security kernel
  – Integrate with hardware to support PLC and HMI
  – Reminiscent of a reusable PC or server motherboard
Example Securing PLC in ICS

- PLC controls the state of output devices
- ICS program executes on PLC trusted device
  - Continuously monitors the state of input devices
  - ICS program makes decisions for control of devices
Illustrative ICS Scan Cycle

- Two modes of operation: Run and Stop
- Typically four stages of run mode
- These will be repeatedly executed
  - Self Scan
  - Input Scan
  - Logic Scan
  - Output Scan

*Automating Manufacturing Systems with PLCs by Hugh Jack*
Decompose Scan Cycle for Security

- Repeating scan cycle is important PLC property
- Not all steps equally critical to ICS
  - Reflect importance in MAC integrity class
- Decompose stages into independent tasks
  - Each performs different task important for PLC
  - Each task will be implemented in different process
  - Assign each task MAC integrity class of importance
  - Trusted Device MAC security kernel fully supports
- Build scan cycle as cooperating processes
  - Use OS synchronization to preserve nature of cycle
Security Techniques

- Use assured pipeline so that each process involved shall communicate with genuine process that it needs to respond or communicate.
- A assured pipeline limits the communication within a sequence of the processes so that each process in pipeline can only receive information to the next process.
- An example would be Aesec virtual Guard Architecture.
Restructure Scan Cycle by Integrity

Diagram:

- **Self Scan:** (low, {diagnostic})
  - Diagnostic Fail
  - Diagnostic Pass

- **Input Scan:** (low, {empty set})
  - Program Execution with provided inputs
  - Input Status

- **Logic Scan:** (medium, {program})
  - Output of the execution
  - Output status: (low, {output status})

- **Output Scan**
  - Critical region
  - Final write to generator: (high, {generator control})

- **HMI:** (low, {output status})
- **Generator:** (high, {generator control})

Cycle Continues
Key Properties of Architecture for CPS

• High assurance MAC to mitigate subversion

• Delivered by reusable Trusted Device for MAC
  – Integrate into high assurance PLC

• Decompose ICS to leverage Trusted Device
  – Each cooperating task designed for a MAC integrity
  – Security kernel enforces integrity MAC (e.g., Biba)
Towards a Highly Secure PLC Architecture for CPS

Nikhil Mahesh,  
University of Southern California

Short Presentation  
Layered Assurance Workshop (LAW)  
Dec 8-9, 2014, New Orleans
Leverage MAC Integrity

<table>
<thead>
<tr>
<th>Subject</th>
<th>Integrity Level</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>Medium</td>
<td>program</td>
</tr>
<tr>
<td>Local HMI</td>
<td>Low</td>
<td>output status</td>
</tr>
<tr>
<td>Remote HMI</td>
<td>Low-Low</td>
<td>output status</td>
</tr>
<tr>
<td>Self Scan</td>
<td>Low</td>
<td>diagnostic</td>
</tr>
<tr>
<td>Input Scan</td>
<td>Low</td>
<td>✴️</td>
</tr>
<tr>
<td>Logic Scan</td>
<td>Medium</td>
<td>program</td>
</tr>
<tr>
<td>Output Scan to HMI</td>
<td>Low</td>
<td>output status</td>
</tr>
<tr>
<td>Output Scan to Generator</td>
<td>High</td>
<td>generator control</td>
</tr>
<tr>
<td>Generator</td>
<td>High</td>
<td>generator control</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC Prg</td>
<td>Program</td>
</tr>
<tr>
<td>Input</td>
<td>General Control, Operation Status</td>
</tr>
<tr>
<td>HMI</td>
<td>General Control, Operation Status, Billing</td>
</tr>
<tr>
<td>Output</td>
<td>Output Status, Diagnostics, Generator Control, Operation status</td>
</tr>
</tbody>
</table>

We need to apply Biba Integrity interpretation, 3 conditions that must satisfy with respect to Subject S and object O are as follows

- Simple Security Condition: S can read O IFF S leq O
- *-Property: S can write O IFF O leq S
- Invocation Axiom: S1 can invoke S2 IFF S2 leq S2