Discrete Control of Response for Cybersecurity in Industrial Control

Gwenaël Delaval, Ayan Hore, Stéphane Mocanu, Lucie Muller, Eric Rutten

Lig, Université Grenoble Alpes, Inria, Grenoble INP

IFAC 2020
Industrial Control Systems (ICS): critical infrastructure

Need for cybersecurity

Control of a response mechanism to potential attacks

Proposal: use of controller synthesis to produce automatically a controller for this response mechanism
Controlled ICS

Industrial control system:

- composed of *Remote Terminal Units* (RTU), connected with sensors and actuators of the physical process
- Programmable Logic Controllers (PLC)
- PLCs and RTUs are connected by a LAN
- PLCs run *programs* controlling the RTUs (possibly several programs by PLC)
Controlled ICS

Industrial control system:

- composed of *Remote Terminal Units* (RTU), connected with sensors and actuators of the physical process
- **Programmable Logic Controllers** (PLC)
- PLCs and RTUs are connected by a LAN
- PLCs run *programs* controlling the RTUs (possibly several programs by PLC)

Attacks on PLCs $\rightarrow$ need for dynamic reconfigurations
Responses to attacks

What kind of response to attacks/alarms?

- Type of attacks considered: **alarms on PLCs**, triggered by an Intrusion Detection System (IDS)

- Dynamic reconfigurations:
  - **isolation** of nodes on the LAN
  - **execution location** of programs on PLCs
  - **execution modes**: Nominal, Degraded, Safe

- Execution modes \(\Rightarrow\) different execution times
Responses to attacks

What kind of response to attacks/alarms?

- Type of attacks considered: alarms on PLCs, triggered by an Intrusion Detection System (IDS)
- Dynamic reconfigurations:
  - isolation of nodes on the LAN
  - execution location of programs on PLCs
  - execution modes: Nominal, Degraded, Safe
- Execution modes $\Rightarrow$ different execution times

Objectives

- execution of programs on non-alarmed PLCs
- keep programs in Nominal or Degraded modes as long as possible
- bound execution time on each PLC
Cybersecurity as a Control Problem

Closing the loop:

- **inputs**: alarms from the IDS
- **outputs**: isolation of nodes of the LAN, modes and execution location of programs
- **state**: current execution modes/location of programs

Combinatorics of solutions $\Rightarrow$ controller difficult to program “manually”
Automation of controller generation: use of Heptagon/BZR

- Managed system modelled as automata and (synchronous) dataflow equations
- Controllable variables defined at runtime by a synthesized controller, to enforce synthesis objectives: invariant temporal properties
- Controller synthesized offline
Heptagon/BZR example

```
node prog(deg,safe:bool) = (e_stop:bool)
let
 automaton
   state Nominal
      do e_stop = false
      unless deg then Degraded
         | safe then Safe
   state Degraded
      do e_stop = false
      unless safe then Safe
   state Safe
      do e_stop = true
end
tel
```

<table>
<thead>
<tr>
<th>deg</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>safe</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>State</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>D</td>
<td>D</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>e_stop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Heptagon/BZR example

node prog(deg,safe:bool) = (e_stop:bool)
let
    automaton
        state Nominal
            do e_stop = false
            unless deg then Degraded
                | safe then Safe
        state Degraded
            do e_stop = false
            unless safe then Safe
        state Safe
            do e_stop = true
    end
end

tel

deg | 0 0 0 1 0 0 0
safe | 0 0 0 0 0 1 0
State | N N N D D S S
e_stop | 0 0 0 0 1 1

define two_progs(deg1,deg2:bool) = (e_stop1,e_stop2:bool)
contract
    enforce (e_stop1 => e_stop2)
    with (safe1, safe2:bool)
let
    e_stop1 = prog(deg1, safe1);
    e_stop2 = prog(deg2, safe2);
tel
Method for obtention of response mechanism controller

Using Heptagon/BZR:

- model PLCs and programs as automata + dataflow equations
- express response objectives as synthesis objectives
- compile and synthesize the controller
Problem stated as:

- a set of $n$ control programs $P_i, i = 1, \ldots, n$;
- a set of $p$ PLCs $C_j, j = 1, \ldots, p$;
- $\max_j$ is the maximum cycle duration of PLC $C_j$;
- $n_{ij}$ is the duration of the nominal version of program $P_i$ on PLC $C_j$;
- $d_{ij}$ is the duration of the degraded version of program $P_i$ on PLC $C_j$. 

Modelling ICS
PLC model

Input: alarm, true when the IDS detects an alarm for this PLC
Output: plc_avail, true when the PLC is “available” (until first alarm)
PLC model

**Input:** alarm, true when the IDS detects an alarm for this PLC

**Output:** plc_avail, true when the PLC is “available” (until first alarm)

Parallel instances for each PLC:

```
plc_avail_1 = plc(alarm_1);
·
plc_avail_p = plc(alarm_p);
```
Program model

- states corresponding to program modes: Nominal (N), Degraded (D), Safe (S)
- input `c_exec_loc`: controllable variable, control the location of the program
**Program model — instances**

Node \( \text{prog} \) instantiated for each program:

\[
\begin{align*}
(\text{mode}_1, & \text{ex}_1, \text{dur}_{11}, \ldots, \text{dur}_{1p}) = \\
\text{prog}^{<<n_{11}, \ldots, n_{1p}, d_{11}, \ldots, d_{1p}>>} (\text{el}_1, \text{cd}_1, \text{es}_1 \text{ or } \text{cs}_1, \text{cw}_1, \text{sw}_1); \\
\vdots \\
(\text{mode}_n, & \text{ex}_n, \text{dur}_{n1}, \ldots, \text{dur}_{np}) = \\
\text{prog}^{<<n_{n1}, \ldots, n_{np}, d_{n1}, \ldots, d_{np}>>} (\text{el}_n, \text{cd}_n, \text{es}_n \text{ or } \text{cs}_n, \text{cw}_n, \text{sw}_n);
\end{align*}
\]

In this instantiation:

- \( \text{el}_i \) are controllable variables for *execution locations* of program \( i \)
- \( \text{cd}_i \) and \( \text{cs}_i \) are controllable variables for switching programs to *degraded* or *safe* modes
- \( \text{dur}_{ij} \) is:
  - 0 if program \( i \) is not executed on PLC \( j \);
  - duration of current mode, if program \( i \) is executed on PLC \( j \)
Global cost model and control objectives

Computation of total duration of programs on each PLC:

\[
dur_{plc_1} = dur_{11} + \ldots + dur_{n1}
\]

\[
\vdots
\]

\[
dur_{plc_p} = dur_{1p} + \ldots + dur_{np}
\]
Global cost model and control objectives

Computation of total duration of programs on each PLC:

\[ \text{dur}_{\text{plc}_1} = \text{dur}_{11} + \ldots + \text{dur}_{n1} \]

\[ \vdots \]

\[ \text{dur}_{\text{plc}_p} = \text{dur}_{1p} + \ldots + \text{dur}_{np} \]

Synthesis objective: cycle duration on PLCs

Duration of execution of programs on PLCs should be less than the cycle time of this PLC

\[ \text{enforce } \bigwedge_{i=1}^{p} \text{dur}_{\text{plc}_i} \leq \text{max}_i \]
Control objectives (contd)

**Synthesis objective: no program on attacked PLCs**

\[
\text{enforce } \bigwedge_{i=1}^{p} \neg \text{plc\_avail}_i \Rightarrow (\text{dur\_plc}_i = 0)
\]

**Synthesis objective: dependencies between safe/emergency stops modes**

\[
\text{enforce } (\text{mode}_i = \text{Safe}) \Rightarrow (\text{mode}_j = \text{Safe})
\]
Control objectives (contd)

Synthesis objective: no program on attacked PLCs

\[
\text{enforce } \bigwedge_{i=1}^{p} \neg \text{plc}_i \text{ avail} \Rightarrow (\text{dur}_i \text{ plc} = 0)
\]

Synthesis objective: dependencies between safe/emergency stops modes

\[
\text{enforce } (\text{mode}_i = \text{Safe}) \Rightarrow (\text{mode}_j = \text{Safe})
\]

One-step optimization: maximize Nominal modes

\[
\begin{align*}
\text{count}_1 &= \text{if } \text{mode}_1 = \text{Nominal} \text{ then } 1 \text{ else } 0; \\
&\vdots \\
\text{count}_n &= \text{if } \text{mode}_n = \text{Nominal} \text{ then } 1 \text{ else } 0; \\
\text{count} &= \text{count}_1 + \cdots + \text{count}_n \\
\rightarrow &\text{ maximize count at each execution step}
\end{align*}
\]
Simulation example

Use-case scenario: 3 programs on 2 PLCs

<table>
<thead>
<tr>
<th>Program</th>
<th>Mode</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alarm2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>critical_wait1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>critical_wait2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>critical_wait3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mode1</td>
<td>Nominal</td>
<td>Safe</td>
</tr>
<tr>
<td>ex_loc1</td>
<td>PLC1</td>
<td></td>
</tr>
<tr>
<td>mode2</td>
<td>Nominal</td>
<td>Degraded</td>
</tr>
<tr>
<td>ex_loc2</td>
<td>PLC1</td>
<td>PLC2</td>
</tr>
<tr>
<td>mode3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex_loc3</td>
<td>PLC2</td>
<td></td>
</tr>
</tbody>
</table>
Scalability

Synthesis time for $n$ programs, running on $n$ PLCs

![Graph showing synthesis time for different numbers of programs/PLCs with and without optimization. The graph plots synthesis time (s) on the y-axis and the number of programs/PLC (n) on the x-axis. The graph includes two lines: one for with one-step optimization and another for without optimization. The y-axis scales logarithmically from 0.1 to 10000, and the x-axis scales from 2 to 7.]
Conclusion

- Approach for the cybersecurity of Industrial Control Systems
- Automated reaction by self-protection to attacks
- Automatically produced controller by controller synthesis
Conclusion

- Approach for the cybersecurity of Industrial Control Systems
- Automated reaction by self-protection to attacks
- Automatically produced controller by controller synthesis

Perspectives

- use of modularity, or hierarchical/distributed controllers to handle scalability
- larger size use-case experiment
- consider possible attacks on communication between the self-protection manager and PLCs