Model-Based Design of a CPS Cyber Security Testbed

Ashraf Tantawy[a], Abdelkarim Erradi[a,*], Sherif Abdelwahed[b]

[a] *Computer Science and Engineering Department, Qatar University, Doha, Qatar*
[b] *Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond VA, USA*

Abstract

This paper presents the design, modeling, and implementation of a modular cyber-physical system (CPS) testbed to conduct cyber security research experimentation for analyzing the security of Industrial Control Systems (ICS). The testbed can also be used for assessing the impact of cyber-attacks on the controlled process and evaluating the effectiveness of attack detection algorithms and defense mechanisms. The paper follows a model-based design approach where the complete testbed architecture is initially presented, followed by modeling each testbed component. The detailed testbed model is then used to demonstrate how to develop more-effective attack experiments that are customized to the given CPS. Finally, a representative attack scenario is presented to show the effectiveness of the testbed and the model-based approach in mounting cyber attacks, simulating physical process hazards, and collecting the necessary data for security analysis.

Keywords: Cyber-Physical System (CPS), CPS Architecture, CPS Security, CPS Testbed, Exothermal Reactor, CSTR, Modeling, Simulation, Model-Based Design (MBD), Process Safety, Safety Instrumented System (SIS), Process Control System.

1. Introduction

Cyber-Physical Systems (CPS) integrate physical elements, for sensing and actuation, with cyber elements, for computation and communication, to automate and control industrial processes. CPS are increasingly used in critical application domains such as health care, traffic management, manufacturing, and energy infrastructures. These systems are increasingly adopting commercial and open source software components and standard communication protocols in order to reduce infrastructure costs and ease integration and connection with corporate networks. However, this has exposed such systems to new security threats and made them a prime target for cyber-attacks to disrupt their normal operation.

*I am the corresponding author

Email addresses: ashraf.tantawy@qu.edu.qa (Ashraf Tantawy), erradi@qu.edu.qa (Abdelkarim Erradi), sabdelwahed@vcu.edu (Sherif Abdelwahed)

URL: http://qufaculty.qu.edu.qa/erradi/ (Abdelkarim Erradi), https://egr.vcu.edu/directory/sherifabdelwahed/ (Sherif Abdelwahed)

This may result in profound and catastrophic impacts such as endangering public safety and economic stability. Despite ongoing efforts to secure and protect CPS, these critical infrastructure components remain vulnerable to cyber attacks. Recent intensified sophisticated attacks on these systems have stressed the importance of methodologies and tools to assess and manage cyber security risks [1].

CPS are widely used in critical infrastructures to monitor and control processes. Experimenting simulated attacks on live systems is often unfeasible or impractical as this may cause undesired system instability. Furthermore, security-relevant datasets from such systems are not publicly available to the research community. Hence, researchers build testbeds that represent scaled down versions of the actual physical industrial control systems in order to experiment with cyber attacks and develop security solutions for such systems. Testbeds provide a research environment that allows extensive attack-defense experimentation with realistic attack scenarios, including vulnerability assessment, risk assessment, impact analysis of cyber-attacks on the controlled process, risk mitigation, in order to enable the design and evaluation of effective detection and defense mechanisms.

Several testbeds developed at various universities and national research labs have been reported in the literature for different CPS application domains, predominantly focusing on critical infrastructures such as the smart power grid testbeds presented in [2], [3], [4]. One of the early power grid testbeds are the National SCADA TestBed (NSTB) [5] and Idaho National Labs (INL) SCADA Testbed which deploys real physical grid components including generators, transmitters, substations and controllers that use industry standard software components and communication protocols [6]. These large scale testbeds provide a high fidelity and realistic operating environment were vendors can assess the cyber vulnerabilities of their latest control systems and evaluate their detection and defense mechanisms. Additionally, aggregate reports are published on common vulnerabilities and best practices to better secure energy networks and assets. Furthermore, these testbeds are also used to improve the standards and for training to increase the understanding of cyber vulnerabilities and associated mitigations. However, most of such testbeds are very costly to implement and maintain, making it unaffordable for most researchers. Furthermore, they mainly use commercial hardware and software components that are often treated as black boxes with little ability to model their inner working. Our testbed follows a model-based design approach with a detailed model of each testbed component. Additionally, it uses open hardware and software components that provide the highest control on the experimental environment and facilitate studying the impact of cyber attacks on the physical process and collecting the necessary data for security analysis.

In addition to the energy domain, a smart transportation system testbed has been proposed in [7]. Industrial Control Systems SCADA testbed has been discussed in [8]. This testbed integrates control systems from multiple critical infrastructure industries to create a functional physical processes to control water distribution, gas pipelines, and manufacturing processes using commercial hardware, software and networking components. A water treatment testbed is reported in [9].

In general, existing testbeds could be classified according to the physical system into two main categories: (1) simulation-based, and (2) operational-based testbeds . Simulation

testbeds rely on process simulation software to generate plant data, without actual physical equipment, sensors, or actuators. The advantage of simulation testbeds is that attack simulation experiments could be performed faster by accelerating the simulator. However, the accuracy of experimental results are limited by the simulator fidelity. Operational testbeds, on the other hand, have physical processes with sensors and actuators, resembling realistic industrial installations. Therefore, they allow more realistic experiments and more accurate conclusions. However, operational testbeds are expensive to build and maintain. As an example operational testbed, the Secure Water Treatment Testbed (SWaT) testbed implements a water treatment cycle with several kinds of filtration using real sensors, actuators and PLCs implementing a complex distributed control [9]. Such operational testbeds utilizing large tanks require significant time to simulate attacks and to detect and visualize their impact. In contrast, researchers at Mississippi State University present a small-scale physical process testbed that uses real sensors, actuators and controllers [8].

As the main focus of our research project is to study the behavior of a variety of cyber attacks and their impact on the physical process, we selected the simulation-based testbed as our design basis. To increase the simulation fidelity, we selected the Continuous Stirred Tank Reactor (CSTR) process equipment, a well-mathematically formulated and well-understood chemical process unit, both in academia and industry. Our testbed distinguishes itself from three different perspectives, which summarize the key contribution of the paper:
(1) **Open System Philosophy**: The testbed uses open hardware and software to allow the highest degree of control on the experimental environment. As an example, we use industrial controllers that run Linux OS and allow low level programming for all software tasks including communication protocols. This is compared to vendor-specific closed-systems that allow only minimum amount of configuration by the end user. Yet, we incorporate industrial standard protocols such as Modbus and DNP3 to mimic actual industrial installations.
(2) **Model-Based Design (MBD)**: We developed and implemented the physical process model, and hardware & software architectures for the testbed, including detailed communication tasks. This allows us to pursue a model-based approach in the experimentation of cyber attacks and the design of security countermeasures. The MBD approach that is followed throughout the research work is formalized in the paper.
(3) **Model-Based Generation of Attack Scenarios**: We use the developed model to illustrate how to systematically design cyber attack experiments that are relevant to the given CPS for more-effective cyber security experimental studies.

The rest of the paper is organized as follows: Section 2 presents the overall CPS architecture and the model-based design approach that will be pursued for the rest of the research work. Section 3 provides a detailed mathematical model for the CSTR process used in the testbed. Section 4 is a complete description of the testbed cyber system, including hardware, software, and communication architectures. Section 5 explains the experimental design of cyber attacks from the developed testbed model, including integrity and Denial of Service attacks. Discussions and reflections are presented in Section 7. Finally, conclusions and future work are highlighted in Section 8.

2. Cyber Physical System Description

Figure 1 illustrates the overall CPS architecture. The physical system is an exothermal continuous stirred tank reactor simulated in real-time using LabVIEW RT Module. Simulation data is exchanged over an I/O physical data bus to both the Basic Process Control System (BPCS) and the Safety Instrumented System (SIS). The BPCS implements the process control functions and runs RT Linux OS. Similarly, the SIS implements the process shutdown logic functions and runs RT Linux OS. A control network interconnects BPCS, SIS, Human Machine Interface (HMI) workstation, and the Engineering workstation. A firewall is configured to isolate the control network from the corporate (Qatar University) network. A DeMilitarized Zone (DMZ) is formed to host the historian and real time data servers, following NIST proposed architecture [10]. Finally, a regulated remote access to the testbed is granted to collaborative researchers at Pisa University, Italy, and Virginia Commonwealth University (VCU), USA, to experiment with remote exploits.

In the rest of the paper, we build a model for each component in the testbed. Our research objective is to pursue a model-based approach to assess the CPS security vulnerabilities as well as to design the security counter-measures. Figure 2 summarizes our model-based research approach. In this paper, we address the first two steps and partially step 3 in the design cycle, as highlighted in gray in Figure 2.

3. Physical Process

We choose the Continuous Stirred Time Reactor (CSTR) as the physical system for the testbed. Reactors play a major role in the process industry. It is an essential equipment in any process plant that generates new products from raw inlet reactants. The CSTR process possesses several features that make it a good choice for CPS security studies. First, the process variables to be controlled are closely-coupled, hence any change in one process variable will impact other variables as well and manifest itself in the overall process behavior. Second, the process has a number of safety hazard scenarios that may be produced by a cyber attack. Finally, mitigation layers for a number of safety hazards rely mainly on the control and safety systems, which are cyber systems that could be compromised by a cyber attack. This last feature highlights a class of CPSs where cyber security is essential for the safe and reliable operation of the physical system.

3.1. Process Description

We consider an irreversible exothermic CSTR process, with a first order reaction in the reactant A with rate k and a heat of reaction λ.

$$\mathrm{A} \xrightarrow{\mathrm{k}} \mathrm{B}$$

Figure 3 shows the Piping & Instrumentation Diagram (P&ID) for the reactor. The reactor vessel has an inlet stream carrying the reactants, an outlet stream carrying the products, and a cooling stream carrying the cooling fluid into the surrounding jacket to absorb the heat of the exothermic reaction. Reactant A enters the reactor with concentration

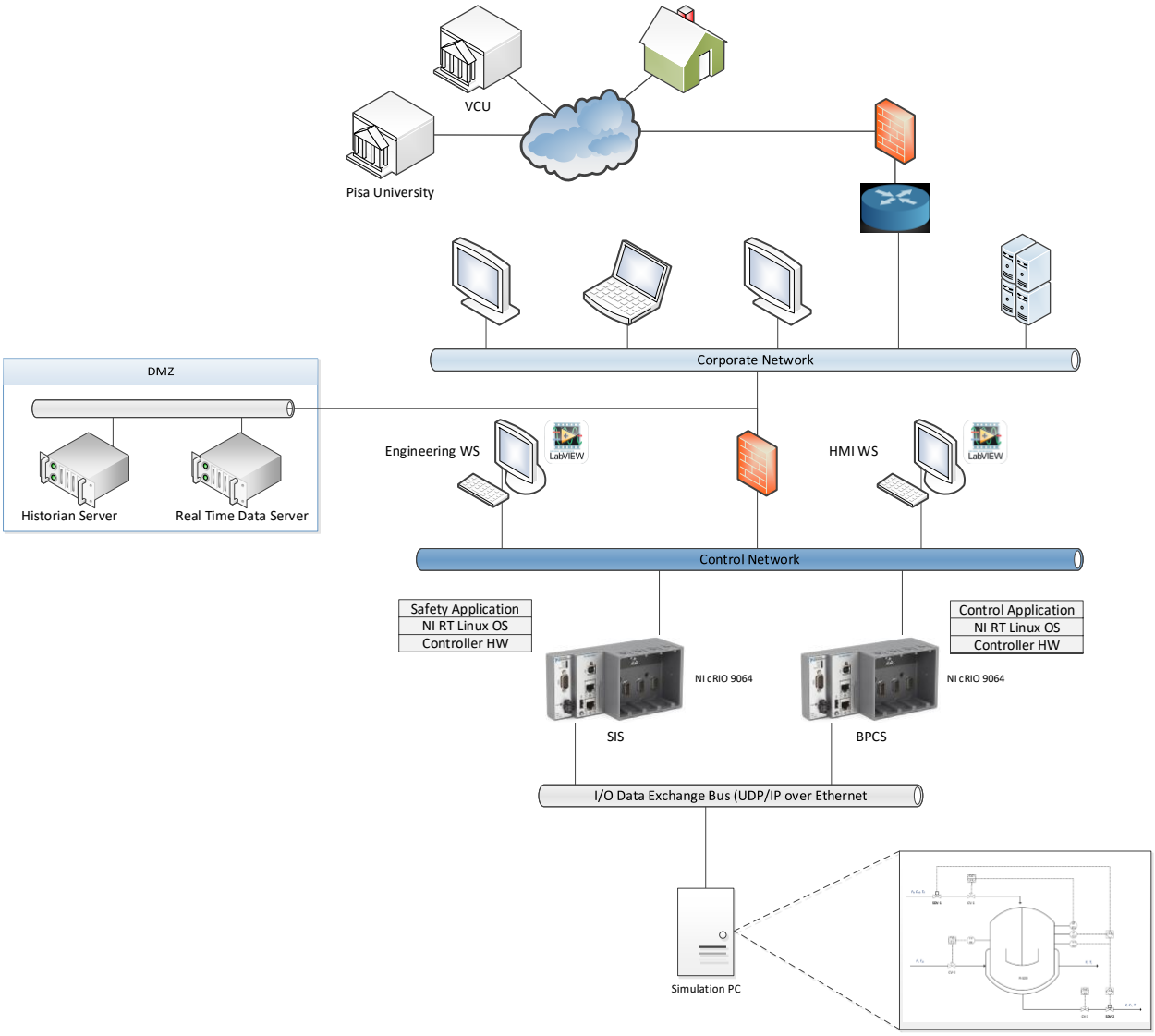


Figure 1: CPS Testbed Architecture

C_{A_0}, temperature T_0 , and volumetric flow rate F_0. A first order reaction takes place where a mole percentage of reactant A is consumed to produce product B. The outlet stream contains both reactant A and product B, with reactant A concentration C_A, outlet temperature T, and flow F. The outlet temperature T is the same as the reactor temperature. The coolant fluid flows into the reactor jacket with temperature T_{J_0} and flow rate F_{J_0}, and leaves the jacket with temperature T_J. The total coolant volume in the jacket is designated by V_J. The control and safety functions shown on the P&ID are explained in Sections 3.4 and 3.5, respectively.

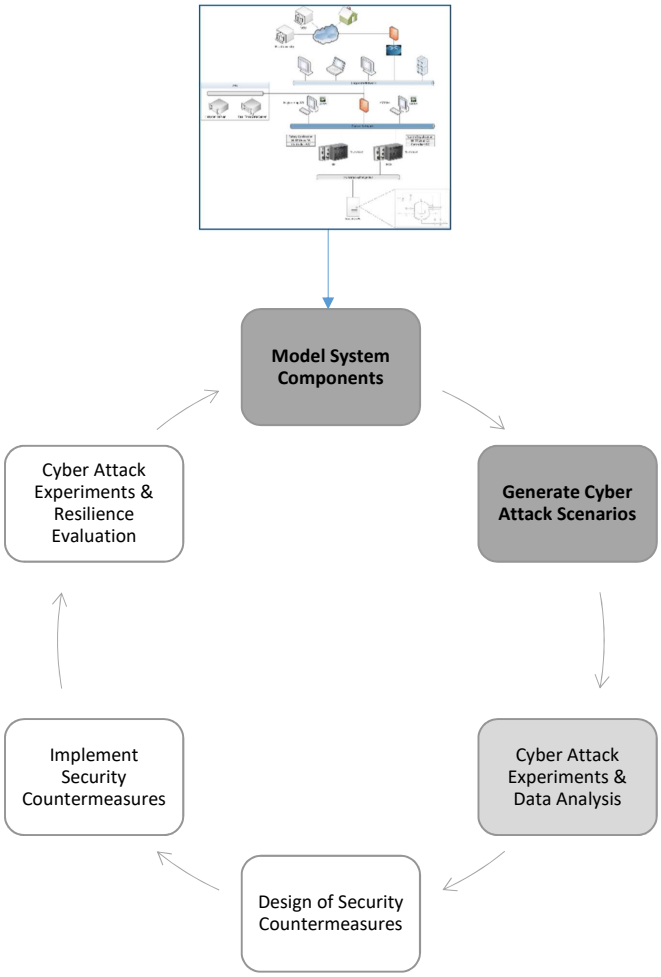


Figure 2: Model-based Design for CPS Security (Research Outlook)

3.2. Mathematical Model

Using fundamental material and energy balance equations, it can be shown that the reactor could be described by the state space equations in Figure 4. The outlet flow, F, is not an independent variable, but varies with the level in the reactor. With a control valve installed at the reactor outlet as per the P&ID, the flow through the valve could be represented by Equation (5), where x is the valve opening and a linear inherent valve characteristic is assumed. To simplify the analysis, the outlet pressure is assumed atmospheric as the reactor. In practice, the discharge pressure is determined by the operating conditions of the downstream process unit. Finally, the valve opening could be controlled manually or automatically to achieve the control objectives. As will be explained in Section 3.4, this outlet valve is not part of the control architecture, and hence it is controlled manually for the sole purpose of regulating the outlet flow to the downstream process.

Table 1 summarizes the model inputs, state variables, disturbances, and parameters. The values shown for the state variables are the steady state open loop values for the indicated

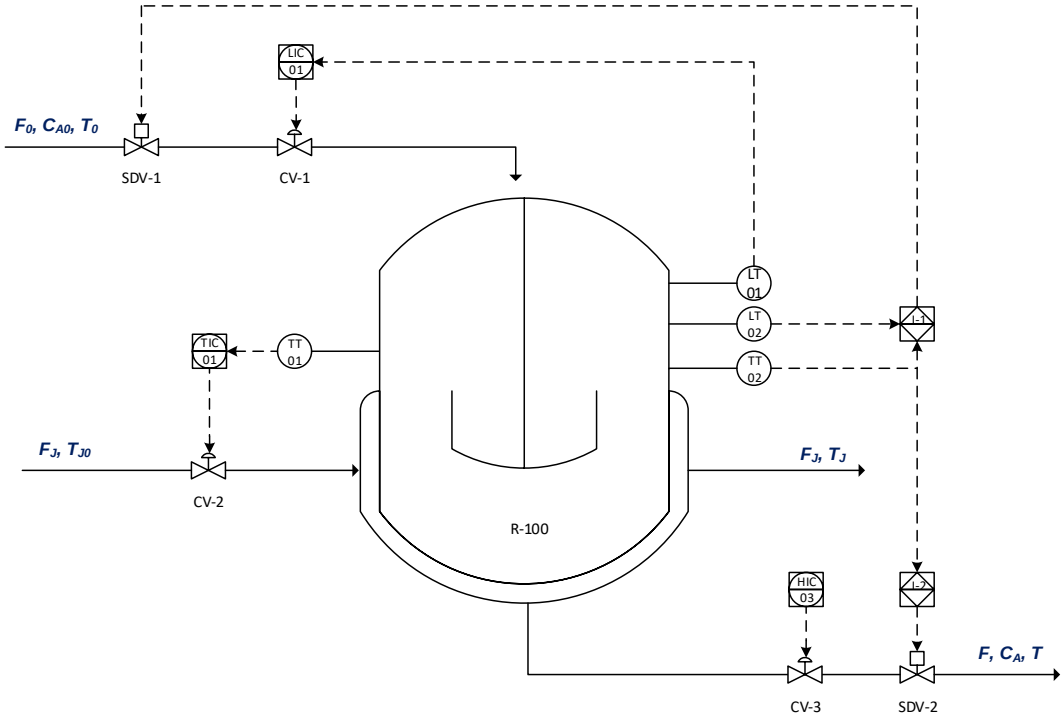


Figure 3: Reactor P&ID

flow inputs. There are 3 steady state operating points. The first and third operating points are stable, while the middle operating point is locally unstable, requiring closed-loop control to maintain the reactor at this operating point. In the rest of the paper, we operate the reactor at the third operating point ($L = 1$ m, $C_A = 15$ moles/m^3, $T = 469$ K, $T_J = 336$ K). For more information about CSTR multiple steady state analyses, refer to [11]. Finally, the values between brackets for the state variables designate the maximum limit for safe reactor operation. The safety system will automatically shutdown the reactor if any of the state variables exceeded the indicated maximum value.

3.3. Process Simulation

The reactor model described in section 3.2 is simulated using LabVIEW Real Time (RT) module running on a simulation server with Windows 10 OS. The simulation uses fixed Ordinary Differential Equations (ODE) solver with simulation step size of 1 ms real-time resolution. As Figure 1 illustrates, the measurement and actuation signals are exchanged with the control and safety controllers using a high speed UDP/IP communication over Ethernet to simulate real time sensor and actuator signals.

The simulation server runs five main tasks via LabVIEW RT. The simulation task solves numerically the model differential equations and runs every 1 msec. Send and Receive tasks for the process controller run every 10 msec to exchange data between the simulation task

$$\frac{dL}{dt} = \frac{1}{A}\left(F_0 - F\right) \tag{1}$$

$$\frac{dC_A}{dt} = \left(\frac{1}{AL}\right)\left(F_0 C_{A_0} - F C_A\right) - \alpha C_A e^{-E/RT} \tag{2}$$

$$\frac{dT}{dt} = \left(\frac{1}{AL}\right)\left(F_0 T_0 - F T\right) + \left(\frac{U A_H}{\rho C_p A}\right)\left(\frac{1}{L}\right)\left(T_J - T\right) + \left(\frac{\alpha\lambda}{\rho C_p}\right) C_A e^{-E/RT} \tag{3}$$

$$\frac{dT_J}{dt} = \frac{F_J}{V_J}\left(T_{J_0} - T_J\right) + \frac{U A_H}{\rho_J C_J V_J}\left(T - T_J\right) \tag{4}$$

$$F = C_v x \sqrt{L} \tag{5}$$

Figure 4: Continuous Stirred Tank Reactor State Space Model

and the BPCS controller. Finally, Send and Receive tasks for the safety controller run every 10 msec and exchange data between the simulation task and the SIS controller.

3.4. Process Measurement & Control

The first key process variables to be controlled is the product concentration, which could be measured directly using an online analyzer. However, this method is expensive and has limited use in the industry, except for very critical equipment in the plant. An alternative approach is to estimate the product concentration from the state space model provided that sufficient measurements exist for the system to be observable. A more practical approach commonly used is to control the concentration indirectly using the reactor temperature. The second key process variable to be controlled is the reactant level to prevent overflow. The liquid level measurement could be easily provided using low-cost level sensors.

From the model equations, the reactor temperature could be controlled by adjusting the coolant fluid inlet flow. A single variable control loop is used to regulate the reactor temperature. The control loop is composed of a temperature sensor TT-01, Temperature Controller TIC-01, and Control Vavle CV-2. PID control algorithm is used for the controller since it is the defacto standard in the process control industry.

The reactor liquid level could be controlled by adjusting the inlet flow rate. A single variable control loop is used to regulate the reactor level. The control loop is composed of a level sensor LT-01, Level Controller LIC-01, and Control Vavle CV-1. Likewise, PID control algorithm is used for the level controller.

Figure 3 shows the two control loops using ISA standard symbols [12].

3.5. Process Safety Shutdown

Process hazards are usually identified through a systematic risk assessment process involving hazard studies, e.g., HAZard and OPerability (HAZOP) study. The four key variables for the process industry are level, temperature, flow, and pressure. For the given process, the operating pressure is assumed atmospheric as there are no rotating machinery.

Symbol	Description	Value [Max. Limit]	Unit	Designation
L	Reactor level	(1, 1, 1) [3]	m	State variable
C_A	Reactant concentration	(1946.81, 1538.9, 15.612) [500]	mole/m^3	State variable
T	Reactor temperature	(318, 350, 469.3) [800]	K	State variable
T_J	Jacket temperature	(303.84, 310.7, 336.226) [800]	K	State variable
F_0	Feed flow	2	m^3/min	Manipulated variable
F_J	Coolant flow	6.163	m^3/min	Manipulated variable
F	Product flow	2	m^3/min	Manipulated variable
C_{A_0}	Reactant inlet concentration	2000	mole/m^3	Disturbance variable
T_0	Reactant inlet temperature	323	K	Disturbance variable
T_{J_0}	Coolant inlet temperature	300	K	Disturbance variable
A	Reactor cross-sec area	1.3	m^2	Model parameter
V_J	Jacket volume	10	m^3	Model parameter
α	Rate of reaction	10^{10}	min^{-1}	Model parameter
λ	Exothermic heat of reaction	1.3×10^5	cal/mole	Model parameter
ρ	Feed density	10^6	g/m^3	Model parameter
ρ_J	Coolant density	10^6	g/m^3	Model parameter
C_p	Feed heat capacity	1	cal/g.K	Model parameter
C_J	Coolant heat capacity	1	cal/g.K	Model parameter
E/R	Activation energy	8330.1	K	Model parameter
$U.A_H$	Heat transfer coefficient	1.678×10^6	cal/K.sec	Model parameter
x	Outlet valve opening	0 to 100%	Dimensionless	Manipulated variable
C_v	Oultet valve flow coefficient	4	m$^{2.5}$.min^{-1}	Model parameter

Table 1: CSTR model variables and parameters

Hazard	**Initiating Event** (Cause)	**Consequences**	**Safeguards** (IPL)
High Level (Reactor overflow)	BPCS failure OR Outlet control valve fully closed OR Inlet valve stuck fully open	2 or more fatalities (safety) Product loss (financial) Environmental contamination (environment)	Reactor dike (Mitigation)
High Temperature (Reactor Meltdown/explosion)	Coolant inlet control valve fully (partially) closed OR Inlet valve stuck fully open	10 or more fatalities (safety) Product loss (financial) Environmental contamination (environment)	None

Table 2: Partial HAZOP sheet for the reactor process

Therefore, pressure and flow hazards, respectively, are not considered. High reactor level will lead to reactor overflow. The hazard of the overflow will depend on several factors, including the toxicity of the reactants, the operating temperature of the reactor, and the occupancy level of operators around the reactor area. Similarly, high reactor temperature will lead to exceeding the reactor design temperature and possible reactor damage. Table 2 summarizes these two key hazards for the process and the possible initiating events.

For the Reactor overflow hazard, inlet stream has to be closed. Following IEC 61511 international standard, both the Safety Instrumented Function (SIF) and the control function have to be independent [13]. Therefore, the SIF will be composed of an independent level sensor LT-02, a logic solver implementing an interlock function I-1, and a feed inlet shutdown valve SDV-1. Upon detecting a high reactor level, the inlet feed will be stopped via the independent shutdown valve.

For the reactor high temperature hazard, the inlet stream has to be closed. Therefore, the SIF will be composed of an independent temperature sensor TT-02, a logic solver imple-

menting an interlock function I-2,, and the inlet stream shutdown valve SDV-1. In addition, since it is preferred to keep the inventory, a shutdown valve SDV-3 is added to the outlet stream. Therefore, upon detecting a high reactor temperature, both the inlet and outlet stream shutdown valves will be closed. These two safety functions are illustrated in the P&ID in Figure 3 using ISA standard symbols [12].

4. Cyber System

Understanding the hardware architecture, the software architecture, and the communication architecture is essential for CPS security design. The hardware architecture provides the possible routes the attacker could pursue to compromise the end controllers in order to cause process hazards. The software architecture gives an in-depth understanding of how the attacker could compromise one or more nodes in the route defined by the hardware architecture in order to launch the required attack. Finally, the communication architecture defines the communication paths and protocols to interconnect different nodes. With this overall architectural view, combined with in-depth knowledge about the physical process dynamics and its associated hazards, as explained in Section 3, a model-based approach could be followed to design an optimal security system for the given CPS according to predetermined optimality criteria, such as the minimization of financial loss or the probability of occurrence of a hazardous event.

4.1. Hardware Architecture

The hardware architecture for the testbed cyber system follows NIST guide to industrial control systems security, using a firewall with DMZ between the control network and the corporate network [10].

4.1.1. Basic Process Control System (BPCS)

The Basic Process Control System (BPCS) executes the control algorithms that implement the control strategy presented in Section 3.4. The testbed uses National Instruments Compact RIO (cRIO) 9064 embedded controller running NI Linux-RT operating system on 667 MHz ARM Cortex-A9 dual-core processor. The controller communicates with the simulator using UDP/IP as described previously. In addition, the controller connects to the control network via an Ethernet interface.

4.1.2. Safety Instrumented System (SIS)

The Safety Instrumented System (SIS) implements the safety shutdown logic presented in Section 3.5. Following IEC 61511 standard, the BPCS and SIS have to be completely independent, including field sensors, logic solver, and field actuators. Therefore, a separate embedded controller is used for the SIS acting as the logic solver. The testbed uses NI cRIO 9063 embedded controller running NI Linux-RT operating system on 667 MHz ARM Cortex-A9 dual-core processor. The controller communicates with the simulator using UDP/IP communication and connects to the control network via an Ethernet interface. This connection enables the exchange of data between the BPCS and SIS. In some recent vendor

implementations, direct SIS interface to the control network is not allowed. In such cases, a dedicated controller acting as a gateway between the SIS and the control network is installed for enhanced cyber security. For the testbed, this gateway function is implemented on the SIS controller using Linux OS security mechanisms. Thus, no separate gateway is required.

4.1.3. Engineering and Operation Interface

One Engineering Workstation is connected to the control network for online software modifications and low level access to the controller hardware. The workstation runs LabVIEW configuration software on Windows 10 OS. Because of the low level access granted to engineering workstations, they represent a major threat to the CPS if compromised. One solution is to connect these workstations only during program download. Unfortunately, most of the industrial plants are dynamic, with a continuous need of small software modifications, signals override, controller hardware monitoring, as well as other functions that require the engineering workstations to be online on a continuous-time basis.

One Human Machine Interface (HMI) is connected to the control network, which acts as an operator interface to the physical process. The HMI reads process information from both BPCS and SIS. In addition, operator actions on the HMI are transmitted as discrete events to the BPCS. It should be highlighted that no operator action is allowed on the SIS, as per industrial standards where the SIS is supposed to run autonomously to minimize the system response delay. The HMI is developed using LabVIEW graphical programming and runs on Windows 10 OS. Figure 5 shows the HMI for the reactor process.

4.1.4. Corporate Network Firewall

Following NIST architectural recommendations, a firewall isolates the control network from the corporate network. The firewall has three network interfaces; the control network, the corporate network, and the DeMilitarized Zone (DMZ). No direct traffic is allowed between the control and corporate networks. Rather, any information exchange takes place via the DMZ. The firewall is implemented using iptables running on Ubuntu Linux [**?**].

4.1.5. The Demilitarized Zone (DMZ)

The DMZ should include all data required by the corporate network, since no direct communication with the control network is allowed. Typically, a historian server and a real time data server are installed. Although both servers receive the same information from plant controllers, there are two major differences; First, the time frame before purging the older data in the database is much longer in the historian server. Second, the real time nature of information gathering for the real-time server imposes more stringent constraints on the selection of the communication mechanism between the controllers and the real time server. The real time data server is implemented using National Instruments LabVIEW Real-Time Module [14]. The historian server is implemented using LabVIEW Datalogging and Supervisory Control (DSC) Module [15].

Finally, the corporate network is Qatar University network and is not part of the testbed. However, the existing network facilities are used for remote access to the testbed.

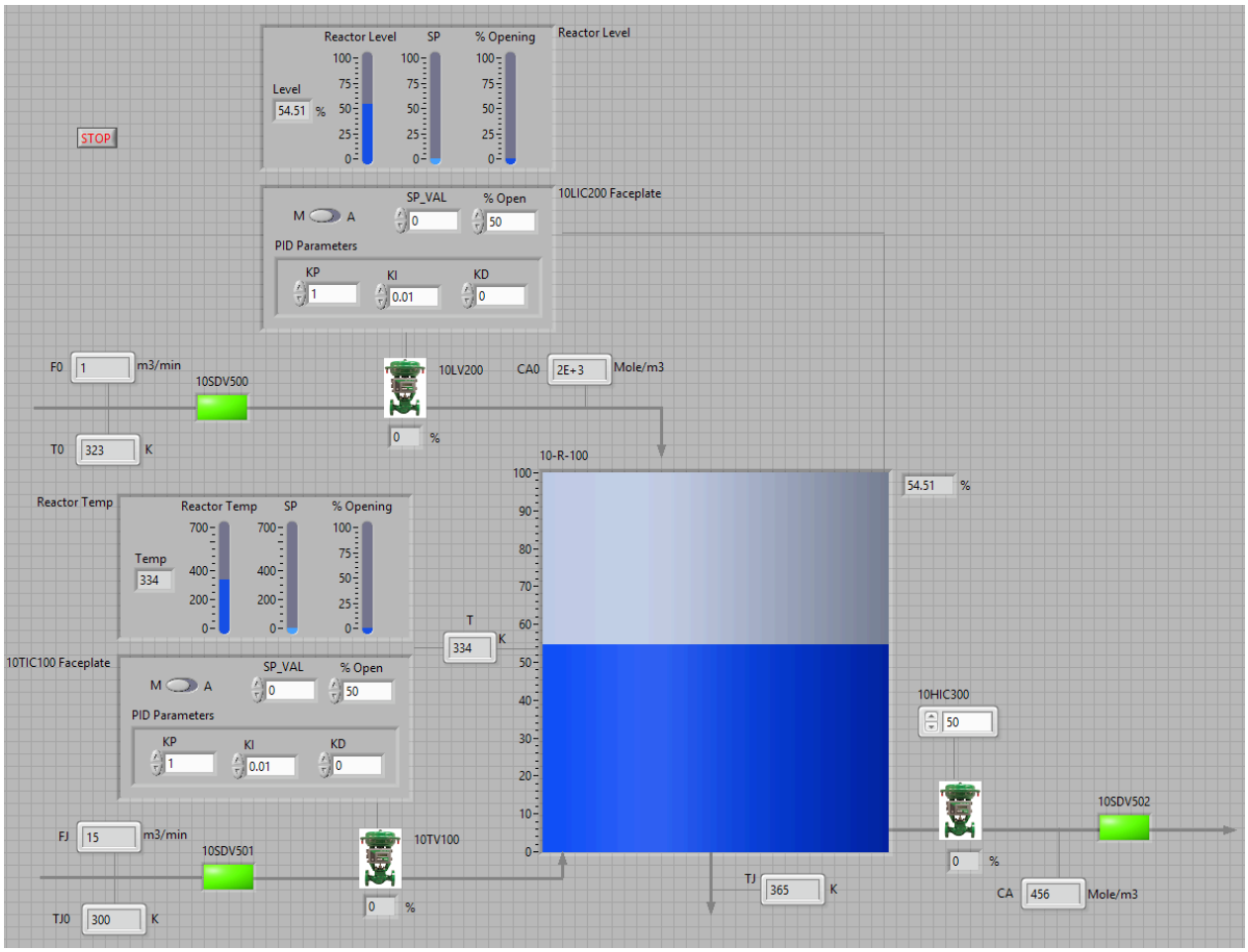


Figure 5: HMI for the Reactor Process

4.2. Software Architecture

The software architecture plays a major role in CPS security. Attacks launched against a CPS are typically multi-step, where the attack propagates from one node to the next exploiting both physical hardware and software connectivities. The software connectivity facilitates the exchange of data between OS processes running either on the same machine (InerProcess Communication-IPC), or on different machines (Inter-Target Communication - ITC). Therefore, understanding the software architecture of the CPS under consideration is crucial for the analysis and design of efficient and cost-effective security systems for the CPS. Fortunately, CPS software architectures for each physical domain (e.g. process, automotive, aerospace industry ... etc.) follow common design patterns, which makes it easier to find a systematic way to analyze and design the security aspects of such systems.

The communication architecture is equally important to understand how an attack could propagate through the system. A complete communication architecture definition includes the communication protocol and the communication mechanisms. We use bare TCP/IP and UDP/IP protocol implementations in the testbed to facilitate the research work. We consider

Communication Mechanism	Determinism	Guaranteed Delivery	High Throughput	Protocol	Loop	Testbed Use
Periodic	Yes	No	Data-dependent	UDP/IP	Periodic	HMI Update
Event-based	Yes	Yes	No	TCP/IP	Event-triggered	HMI Commands, SIS async data
Network stream	No	Yes	Yes	TCP/IP	Periodic	Historian update

Table 3: Testbed communication mechanisms

three communication mechanisms: (1) periodic communication, where the receiving node is interested only in the last value, hence determinism is important while guaranteed delivery is not as critical, (2) Event-based communication, used mainly to transfer asynchronous data such as HMI commands, where both guaranteed delivery and deterministic performance are required, and (3) Data stream communication, where a large amount of data is streamed over a communication link for logging purposes, hence guaranteed delivery is important, while determinism is not as critical. Table 3 summarizes the three network communication mechanisms along with their use in the testbed.

In designing the testbed, the system tasks including control tasks, safety tasks, and data communication tasks, are divided into separate software processes (modules) to allow for design flexibility, scalability, and robustness against software crashes. Processes on the same target communicate using IPC while processes on different targets communicate using ITC. The choice of the data communication mechanism for each path depends on the data transfer requirements, among other factors. Sections 4.2.1 to 4.2.3 explain the processes for each node and the relevant data communication mechanisms.

4.2.1. Basic Process Control System (BPCS)

Figure 6 shows the software architecture for the BPCS and its connectivity with other testbed nodes. The control and communication tasks are split into different processes to avoid the non-determinism that may be introduced by the communication processes from impacting the critical control functions.

The core tasks of the BPCS are the control algorithms. These comprise the PID controllers as well as other custom control tasks such as sequential control or control logic required for more sophisticated control strategies. Control tasks receive discrete commands from both the HMI and the SIS over the control network using discrete-event communication. Control tasks also exchange field input/output data with the physical process visa periodic communication.

Finally, all physical process data is transmitted to the historian server for logging purposes and to the real-time server for online monitoring and analysis.

It should be highlighted that the HMI is not allowed to write directly to field devices with this software architecture. Operators utilizing the HMI are allowed to manipulate field data only through the functionality built into the control tasks.

4.2.2. Safety Instrumented System (SIS)

Figure 7 shows the software architecture for the SIS. The core functionality is represented by the Safety Logic task implementing the combinational and sequential logic required for the safe process shutdown. In practice, the safety logic is a complex large program that safeguards an entire plant if the process variables exceeded their safe operating limit. Therefore,

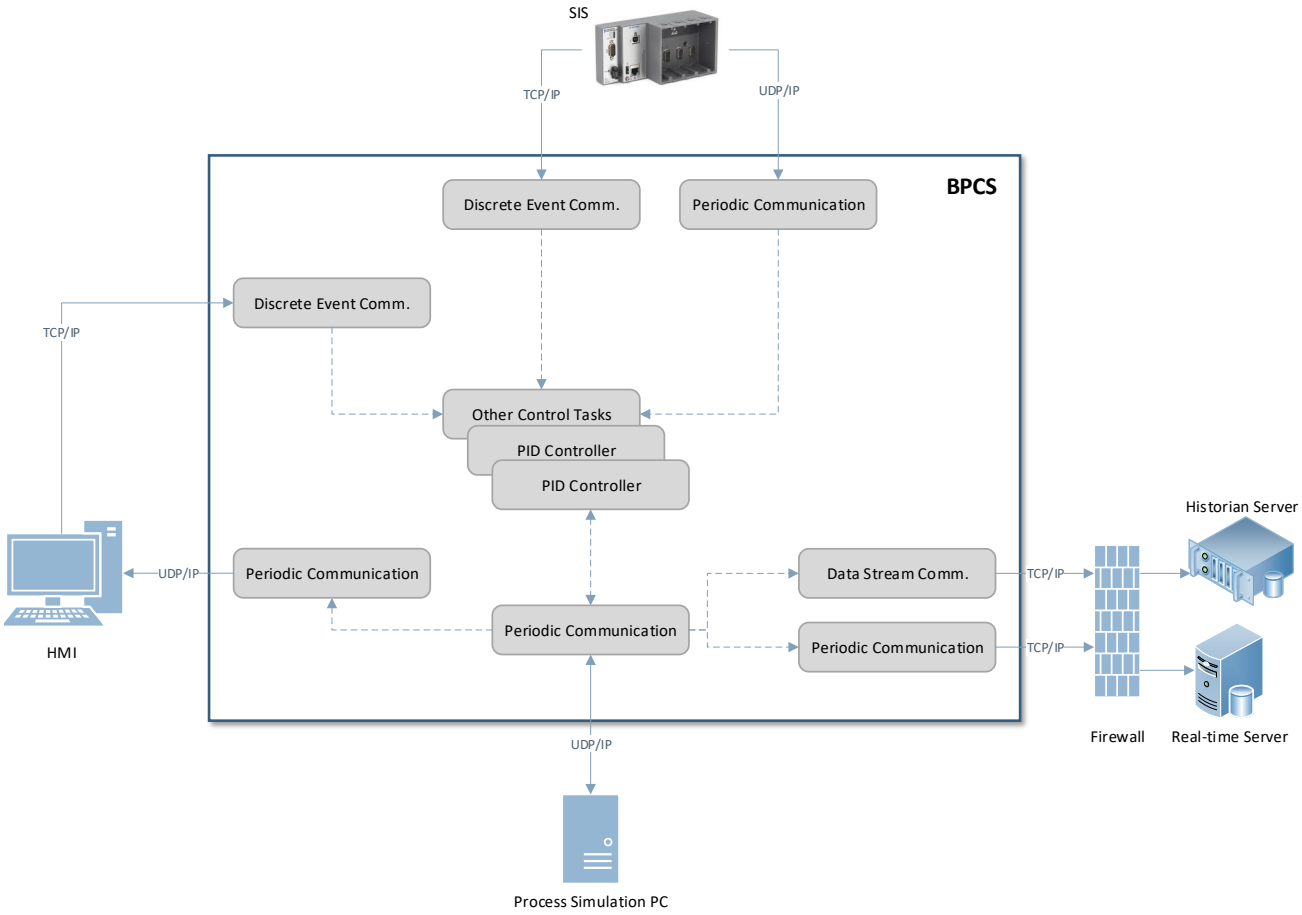


Figure 6: BPCS Software Architecture

it is typically divided into a number of communicating subtasks that follow the modular design approach.

The Safety Logic communicates sensor and actuator data with the actual plant. For the testbed, a periodic communication task exchanges the physical system data with the process simulator. The SIS communicates two types of data to the BPCS. First, field measurements and final element status is reported periodically to the BPCS. A periodic communication task is used for this purpose. Second, discrete events that take place during a shutdown are communicated to the BPCS for display purposes and for further control actions. A discrete event communication mechanism is used for this infrequent data exchange. It should be noted that all communication for the SIS is outgoing to other control system nodes, and that there is no incoming communication. As an example, the SIS does not have direct communication to any HMI for safety reasons. Operators cannot interact directly with the SIS to alter its function since the SIS is designed to be autonomous. Also, to display SIS information on the HMI, data is passed first to the BPCS through the two aforementioned communication mechanisms. Similarly, BPCS does not have a communication link with SIS to transmit information.

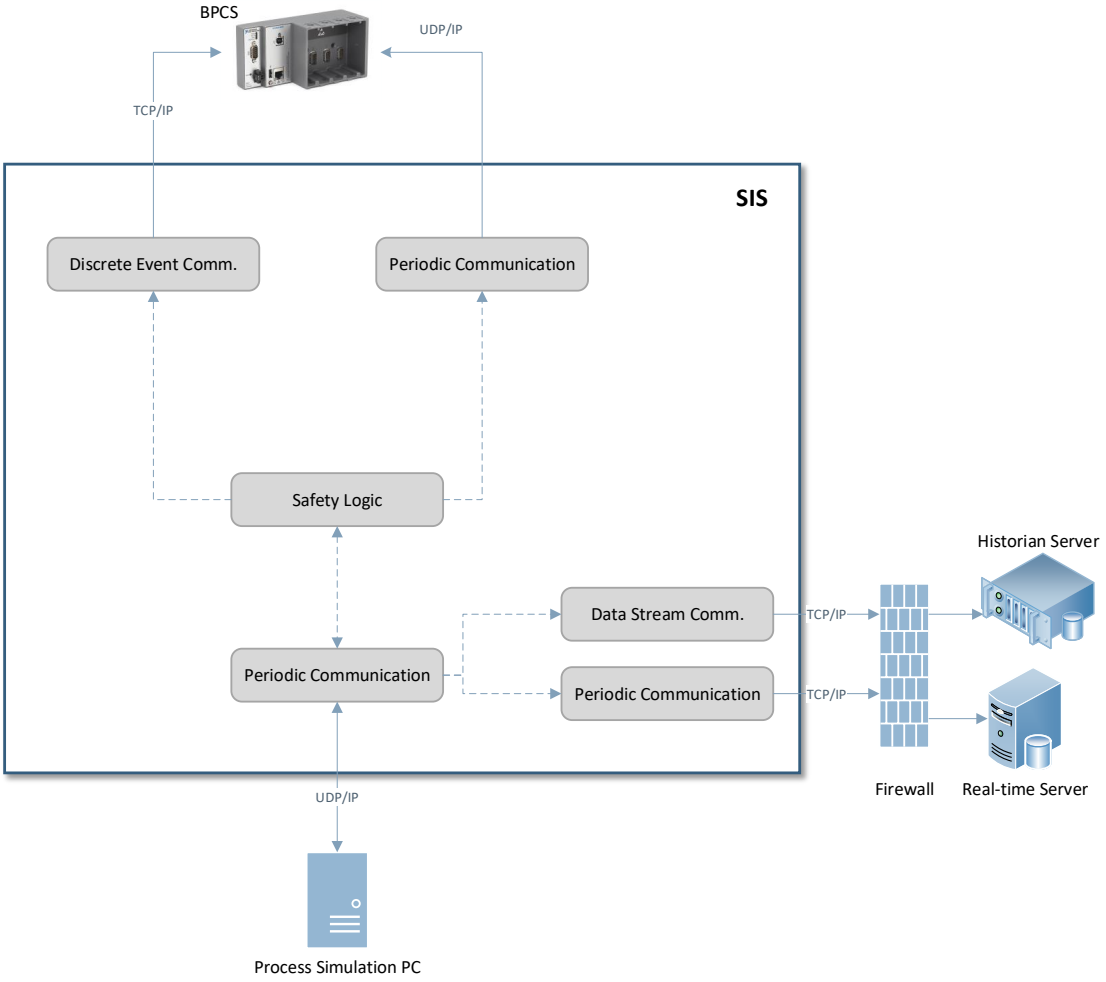


Figure 7: SIS Software Architecture

Although this architecture is the best approach to minimize the probability of SIS compromise, it is not always followed by the industry due to the operability constraints it imposes on the system. As a real-life example, upon triggering a shutdown action by the SIS, the SIS is typically required to submit a command to the BPCS to execute further control actions. In such scenario, the SIS requires a feedback signal that the control actions were successfully executed by the BPCS. Otherwise, further shutdown actions may take place. Similar scenarios are often encountered in the process industry, and therefore one way communication from SIS to BPCS may not be feasible in all cases.

Finally, one way communication links are established between the SIS and both Historian and Real time servers, akin to the BPCS, for logging purposes.

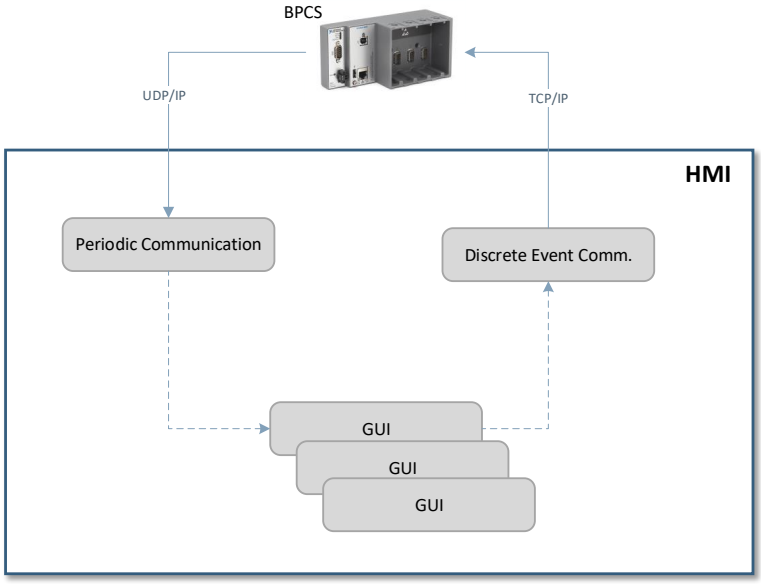


Figure 8: HMI for the Reactor Process

4.2.3. Human Machine Interface (HMI)

Figure 8 shows the software architecture for the Human Machine Interface (HMI) workstation. The main task for the HMI is to provide a Graphical User Interface (GUI) for the operator to monitor and control the plant. The HMI has access control mechanisms that allow different levels of control based on the user role (typically operator, supervisor, and engineer). Despite the level of access, only the objects that are configured in the HMI software could be controlled. For example, the engineer cannot modify a PID controller through the HMI if it is not configured by the HMI software as a GUI object. This fact becomes important when designing the security protection layers for the CPS. The more limited functionalities available in the GUI, the less opportunity there is to launch an attack against the process via the HMI node.

The GUI tasks receive the plant data periodically from the BPCS via a periodic communication task. Simultaneously, the GUI transmits the operator commands via a discrete event communication task to the BPCS for execution. As stated previously, the HMI does not have a communication link with the SIS to ensure plant safety.

4.2.4. Engineering Workstation

Finally, the engineering workstation hosts the programming tool for the controller. This workstation usually has full access to controller hardware and software resources. We assume that this workstation is connected only to the network when there is a need to modify or reconfigure the controller, and hence it is not relevant for security studies. As mentioned previously, many plants in practice connect the engineering workstations continuously to address the day to day modification needs. As this practice may increase the security threat

for the CPS, we will consider it in future work.

Figure 6 shows the complete software architecture for the cyber system depicted in Figure 1.

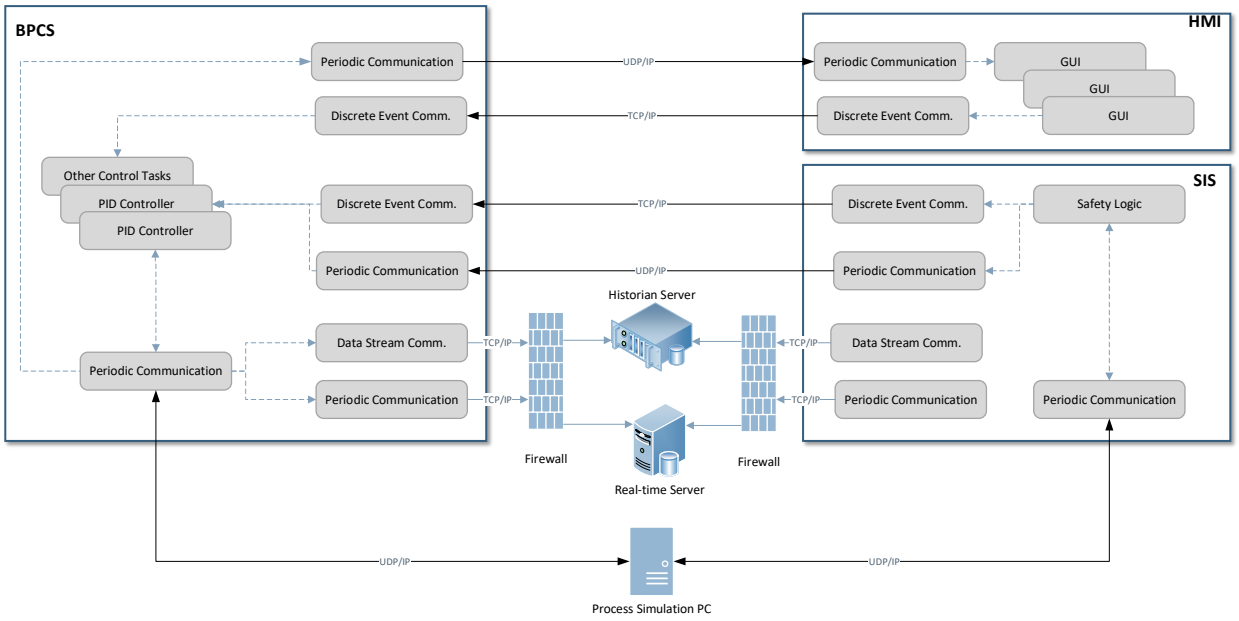


Figure 9: CPS Testbed Software Architecture

5. Experimental Design of Cyber Attacks

Availability is one of the important objectives CPS design strives to maintain via several techniques such as fault-tolerance and diversity techniques. System unavailability may cause a range of consequences, from financial loss to system damage and loss of life. CPS unavailability maybe caused by cyber attacks such as a Denial of Service (DoS) attack.

Another important CPS design objective is integrity, which is the assurance that information and programs are changed only in a specified and authorized manner. Any data manipulation may bring the CPS to an unsafe state. Given the inter-connectivity between CPS nodes, data manipulation could propagate from one node to another, causing simultaneous malfunctioning of different protection layers, and consequently a major damage for the CPS.

One of the advantages of model-based design is that attack experiments and penetration testing could be designed specifically for the system under consideration, as opposed to either random tests or comprehensive time consuming tests. Given the detailed hardware and software architectures for the testbed cyber system presented in this paper, attack experiments could be rigorously designed to test the system against DoS and integrity attacks. For the rest of this section, the way attacks propagate through the testbed is initially de-

scribed, then the design of DoS and integrity attack experiments is presented. This section represents a roadmap for further research that will be conducted on the testbed.

5.1. Attack Propagation

The ultimate objective of any CPS attack, whether integrity or DoS, is to harm the physical process by compromising either the BPCS or the SIS, or both. The compromise could be in terms of manipulation of sensor and actuator data, as in integrity attacks, or in terms of malfunctioning of the controller, as in DoS attacks. To achieve this target, the attacker does not have to compromise these nodes directly to mount an attack. This fact can be illustrated easily by inspecting the overall software architecture for the CPS. In order to mount an attack, it is sufficient to compromise any task or communication link in the data path that leads to the BPCS or SIS. The attacker then injects the malicious data or floods malicious traffic into the normal data stream, where eventually the data will reach the target node.

In order to enumerate the possible attack points, a data flow diagram could be extracted from the software architecture in a straightforward manner. As an example, Figure 10 shows the data flow diagram for all possible paths that lead to the BPCS. The diagram starts at the attack entry point, which could be either an outside attacker PC over the Internet or a corporate LAN PC in case of an insider attack. The outsider attacker will eventually need to compromise a corporate LAN PC to gain access to the rest of the network. From the corporate LAN PC, there are two routes to access the control network nodes. The first route is directly through the firewall, which may be difficult to pursue as the firewall does not allow direct communication between the corporate LAN and the control LAN. The second route is via the Historian / real-time servers. This route is easier to achieve, as the communication from corporate LAN to servers, and from servers to the control LAN is a legitimate route that is allowed by firewall rules. The mechanism by which the attacker will gain access to the control LAN via either route follows typical IT vulnerabilities and will not be discussed in the paper.

For the control LAN side, the diagram shows the processes running on each node and the communication links between them. Each task is labeled according to the physical node where it is running, and each communication path is annotated with the type of communication (whether IPC or ITC) and the communication protocol used.

From the data flow diagram, an attack tree could be derived. Figure 11 shows a high-level attack tree extracted from the BPCS data flow diagram in Figure 10. This attack tree shows two types of abstract events for each attack path to cause a process hazard; a basic event (circled), and a compromise event. The attack tree does not have information about how the basic event or the compromise event could take place. Therefore, the tree could be constructed at early stages of the design where sufficient knowledge about the hardware and software is not yet available. As the design get refined, each of the abstract events could be further expanded to enumerate the possible causes according to the node hardware and software specifications. For example, Figure 12 shows an attack tree to compromise the HMI machine given its specification as a PC running Windows 10 OS. This attack tree represents the "HMI Compromise" circle in Figure 11.

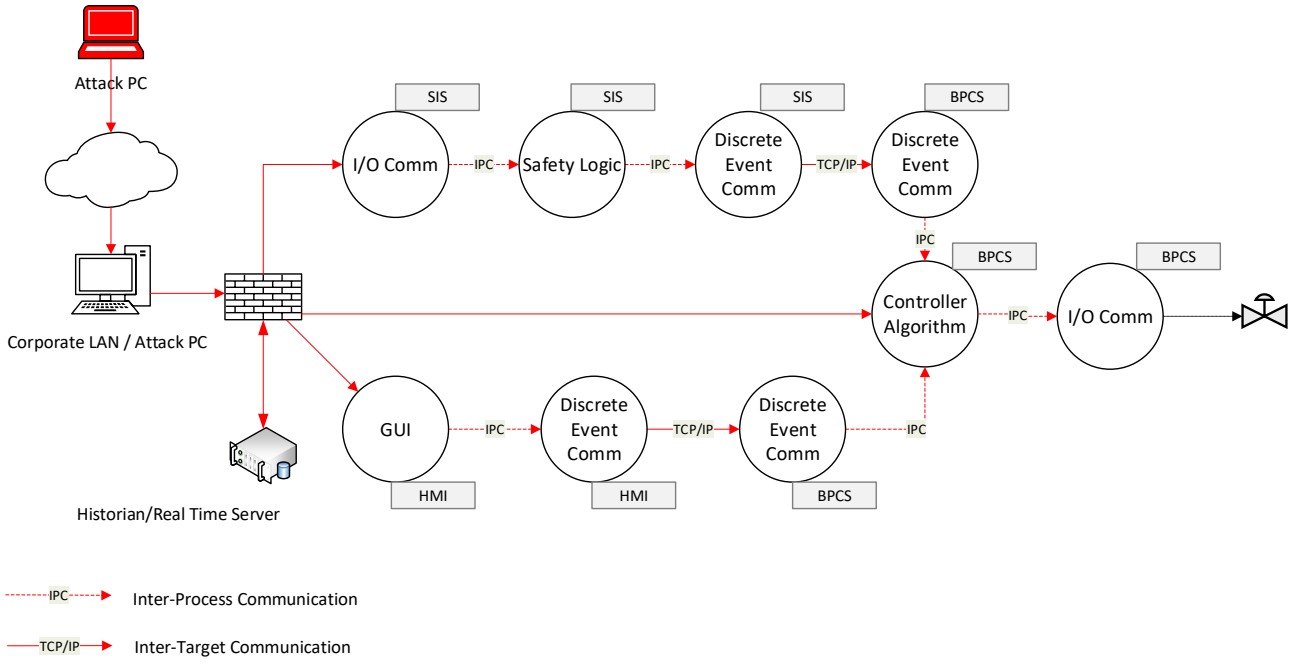


Figure 10: BPCS Cyber Attack Data Flow Diagram

Once one node on the control network is compromised, an integrity attack or DoS attack could be launched, depending on the type of malicious traffic injected.

It should be highlighted that the BPCS attack tree in Figure 11 starts from control network nodes, with the underlying assumption that an access has already been obtained to a control network node. The pre-requisite to gain elevated access to the control network either locally or remotely via a remote session has been studied extensively in the IT Security literature and will not be treated in the paper. For further details about IT security vulnerabilities, refer to [16].

5.2. Design of Integrity Attack Experiments

Integrity attacks aim to manipulate the data of field sensors and actuators. Some attacks may be as simple as random manipulation of field data that could be detected easily either from other sensor measurements or from the system model. Other attacks may be sophisticated enough to deceive the operators and engineers via manipulation of HMI values as well as system observers.

An integrity attack could be mounted by manipulating the payload of any network stream that leads to the target node. As an example, Figure 11 shows the BPCS attack tree illustrating the possible ways to inject malicious data either by manipulating the process that generates the data, e.g., HMI GUI task, or by compromising the data communication task that interfaces with the network, e.g., SIS discrete-event communication task.

The design of the attack experiments is just an enumeration of all possible paths on

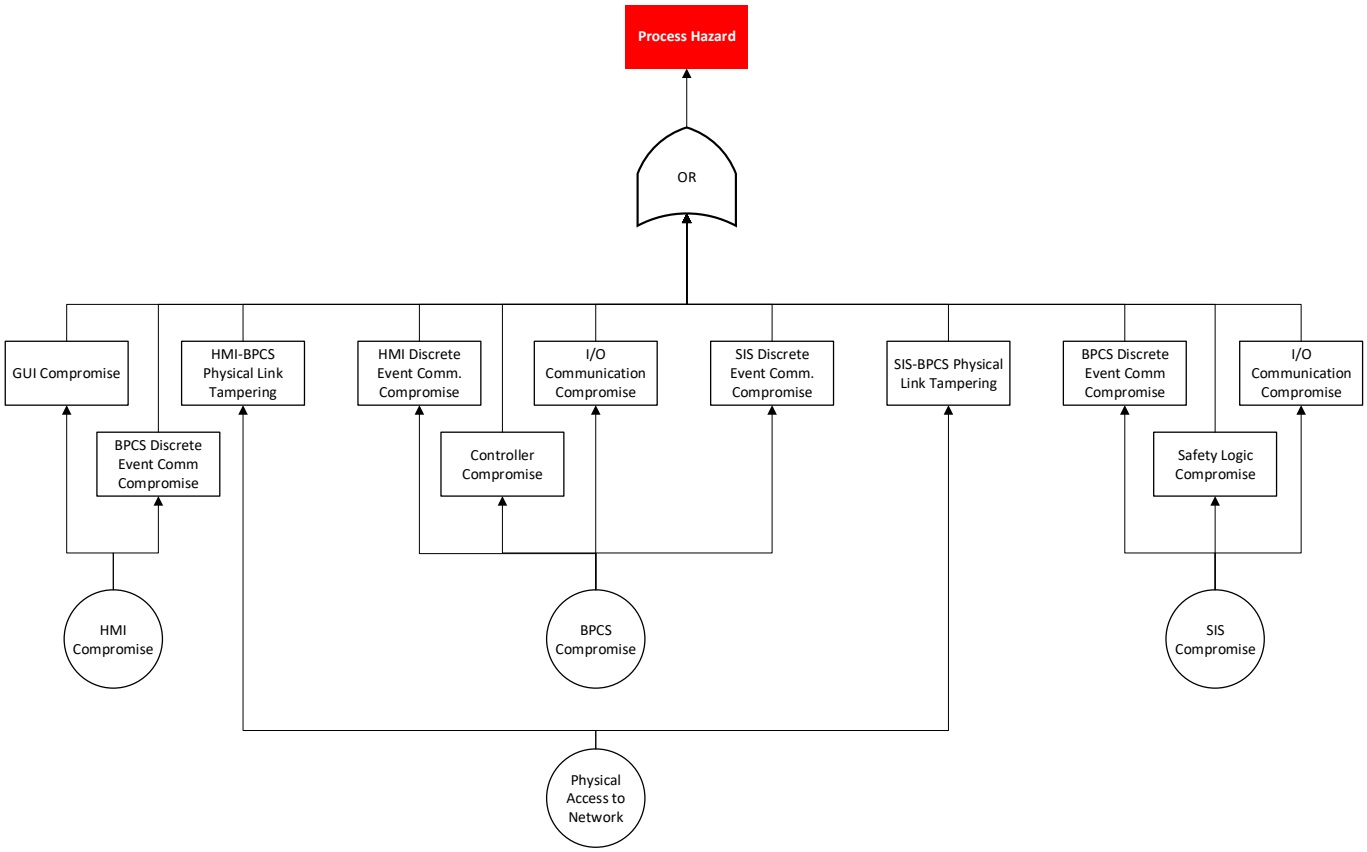


Figure 11: BPCS Attack Tree

the attack tree, instantiated with the specific compromise type of each node in the attack tree. For example, Figure 13 shows the design of one experiment for BPCS integrity attack through the compromise of the HMI workstation. The design of the experiment is derived from both Figure 11 and Figure 12.

All other experiments to attack the BPCS could be designed in a similar way. The detailed design for each experiment is omitted for brevity.

This example shows how an attacker could exploit the strong hardware and software connectivity features of most cyber physical systems, whereby it is sufficient to compromise one control network node and then exploit CPS connectivity to reach the target node.

5.3. Design of DoS Attack Experiments

The two main functions provided by the CPS are: (1) Physical system control, using the BPCS, and (2) Physical system safe shutdown, using the SIS . DoS attack that causes malfunctioning of the BPCS will eventually result in a safe shutdown of the physical system using the SIS upon significant deviation of the process variables. This will incur financial loss only, and possibly minor environmental impact (e.g., depressurization of the chemical product to the atmosphere via the flare system). DoS attack against the SIS, either individually or simultaneously with a DoS against the BPCS, will cause a physical system hazard

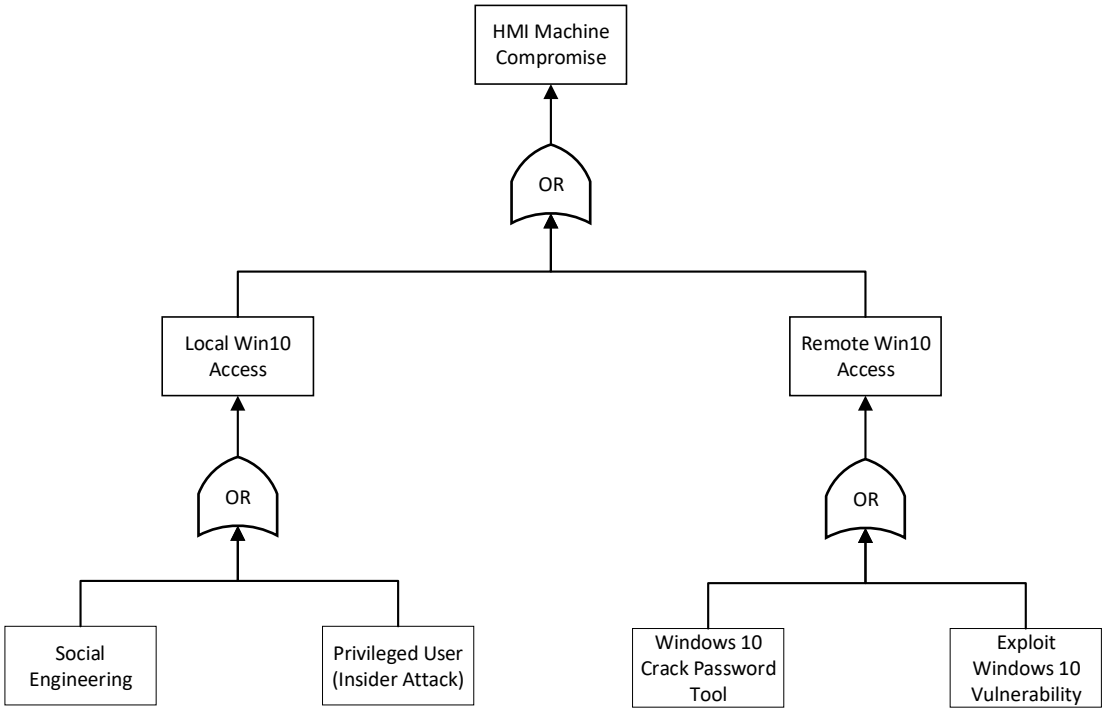


Figure 12: HMI Compromise Attack Tree

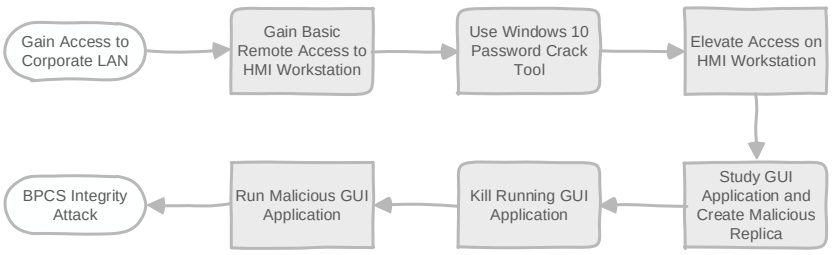


Figure 13: Experiment for a BPCS Integrity Attack

only if it caused a SIS dangerous failure. The dangerous failure represents the situation where the SIS does not safely shutdown due to its malfunction. An example of this failure caused by DoS attack is when the SIS processor is not able to calculate the correct output value to the field actuator due to a very high load on the SIS Node. However, the ability of a DoS attack to cause a dangerous failure for the SIS depends heavily on the SIS design. The SIS is usually a sophisticated hardware that is equipped with both hardware and software monitoring sensors, e.g. watchdog timers, that detect the malfunctioning of the processor and I/O hardware and enforce the shutdown if necessary. The possibility of DoS attacks to cause a dangerous failure of the SIS is an interesting research problem to explore the possible

attack scenarios and their countermeasures. In this section, we present only the design of the attack experiment that lead to a DoS attack to testbed nodes, and leave the impact of the DoS on the SIS for future work.

Assuming the BPCS is the target node for a DoS attack, Figure 11 still represents the possible ways to launch a DoS attack against BPCS. Any task that could be compromised, either a data-generation task or a communications task could be used to flood the BPCS with network data. Depending on the design of the BPCS and its robustness to communication errors, excessive data communication activity could result in a communication halt, processor high utilization that will impact process monitoring, or a complete crash of the BPCS.

Similar to integrity attacks, the design of DoS attack experiments is just an enumeration of all possible paths on the attack tree, instantiated with the specific compromise type of each attack tree node. Figure 14 shows the design of one experiment for BPCS DoS attack through the compromise of the HMI workstation.

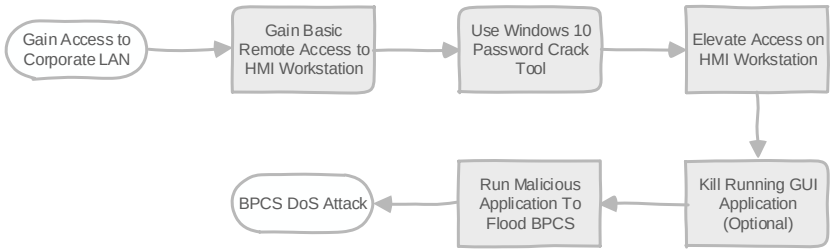


Figure 14: Experiment for a BPCS DoS Attack

It should be noted that the design of the experiments for both integrity attacks and DoS attacks is very similar for each node. The difference between a DoS attack and the integrity attacks lies in the attacker's action post node compromise. A DoS attack is easier since it does not require knowledge about either the physical process or the software task running on the controller (hence killing the GUI is optional in the DoS attack as shown in Figure 14). On the other hand, an integrity attack is more sophisticated as it requires thorough understanding of the data generation and network communication protocol details in order to inject malicious data. Furthermore, physical process knowledge is required for concealed attacks.

6. Testbed Experiments

In this section, we present three attack experiments conducted on the testbed following the experimental design discussed in the last section. Experiment 1 is a DoS attack against the BPCS only, with no attempt to manipulate the process variables (no integrity attack). Experiments 2 and 3 are multistage attacks, where initially an integrity attack is launched to manipulate the process in such a way to cause a safety hazard, followed by a simultaneous DoS attack against BPCS and SIS to disable both the operator and the safety system to take

any corrective action. Experiment 2 involves a reactor overflow hazard, while Experiment 3 causes a reactor runaway hazard.

6.1. Experiment 1 - BPCS DoS Attack

A DoS attack was launched against the BPCS controller from the compromised HMI workstation. We assume that the HMI has already been compromised and that the attacker is able to run a malicious software (i.e., tasks in the first row of Figure 14 have been successfully completed). Furthermore, all testbed components are running normally and the physical process is stabilized at the beginning of the simulation time, $t = 0$. The conducted experiment was as follows:

- $t = 0$: An ICMP flood DoS attack was launched from the compromised HMI against the BPCS at simulation time $t = 0$, using hping3 penetration testing tool and the following command:

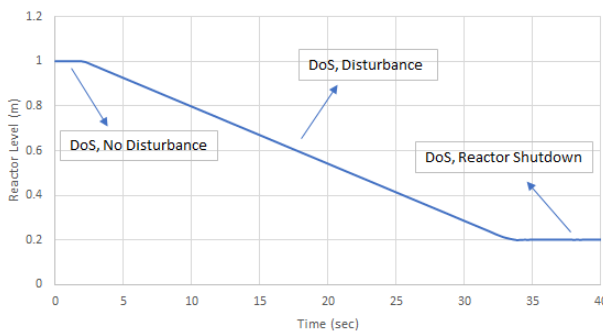
$$\text{hping3 -1 –flood –spoof} \quad \underbrace{192.168.0.2}_{\text{HMI}} \quad \underbrace{192.168.0.5}_{\text{BPCS}}$$

- $t = 2$ sec : A process disturbance was injected in terms of a sudden increase in the demand at the reactor outlet by an amount of 2 m^3/min. Therefore, the reactor outlet flow exhibited a step increase. The four reactor state variables were recorded.

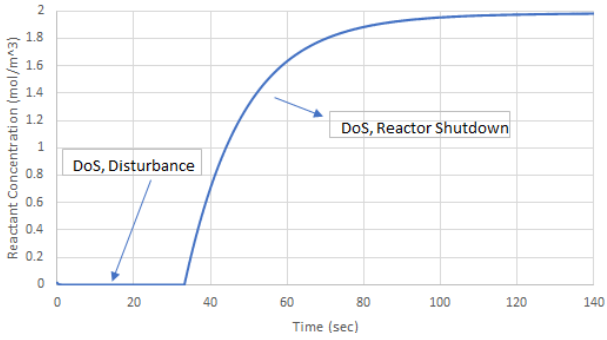
Figure 15 presents the reactor state variables during the experiment.

- $0 < t < 2$ sec : Following the ICMP command launch, the BPCS controller stopped responding to any communication with other testbed nodes, including the Simulator server, in about one second. The HMI software reported a loss of communication with the BPCS. Since the process was initially stable, the reactor did not go out of control for the first 2 seconds prior to introducing the outlet flow disturbance. This is mainly because the outputs from the BPCS hold their last values when the BPCS fails to update its outputs. Therefore, the process control valves stayed in their last position that stabilized the process prior to introducing the disturbance. Accordingly, the four reactor variables in Figure 15 have the same steady state values reported in Table 1.

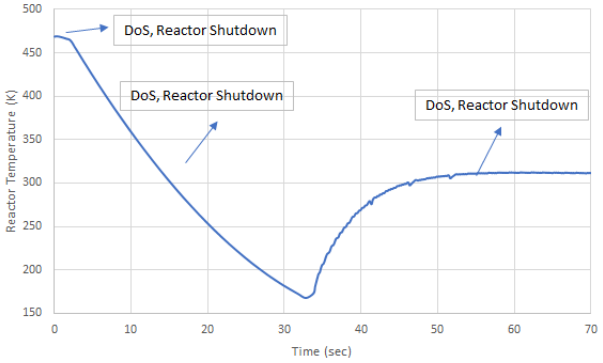
- $2 < t < 34$ sec : Following the disturbance, the reactor level cannot be maintained with the same reactor inlet flow and increased outlet flow due to the loss of BPCS control. The reactant level decreases linearly with time until it reaches the minimum allowable limit of 0.2 meters at $t = 34$ sec, at which time the SIS triggers a reactor shutdown by closing the inlet and outlet shutdown valves. The reactant concentration did not exhibit a significant change during the time the level was steadily decreasing. The reactor temperature drops rapidly due to the increased outlet flow, where the temperature drops below the zero degree Celsius level. Depending on the reactor design and the material of construction, this rapid drop in temperature can cause material fatigue and may lead to fractures. Similarly, the coolant temperature drops due to the decreased reactant temperature.



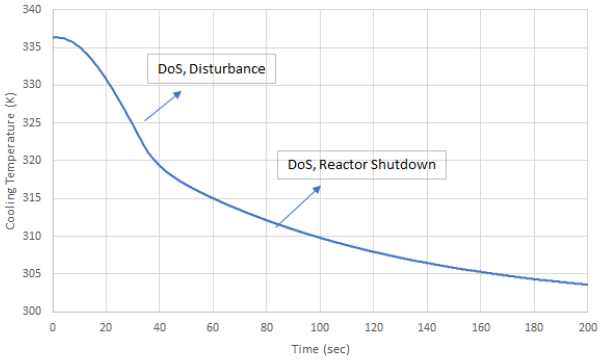
(a) Reactor Level



(b) Reactant Concentration



(c) Reactor Temperature



(d) Cooling Temperature

Figure 15: Experiment 1: BPCS DoS Attack with a Reactor Consumption Disturbance

- $t = 34$ sec : The SIS is triggered and the inlet and outlet shutdown valves (SDV-1 and SDV-2, respectively) are forced to the closed position.
- $t > 34$ sec : After the reactor shutdown, the reactant level stabilizes at 0.2 m. The reactant concentration increases significantly due to the fact that the stagnant reactor inventory is continuously cooled down and hence the reaction takes place at lower temperatures. The reactant concentration reaches its steady state value after about 140 sec. The reactor temperature recovers from the subzero level due to the increased reactant mole concentration. The coolant temperature continues to drop until it reaches a steady state around 300 K after more than 200 sec.

This experiment shows that the process may not go out of control immediately after a BPCS attack. If the process is stable and there are no disturbances, only operators will be affected since the HMI will loose communication with the BPCS. Although most processes have continuous disturbances during normal operation, the true impact of the DoS attack will depend on the magnitude of the disturbance and the sensitivity of the process to the active disturbance during the attack.

The SIS acted as the last protection layer to safeguard the reactor. If the DoS attack targeted both the BPCS and SIS simultaneously, the reactor would have gone out of control and the reactor temperature would have increased dramatically, possibly causing a reactor

meltdown. This result shows the significant importance of protecting the SIS from BPCS cyber attacks for the class of hazards that do not have non-electronic protection layers, such as the reactor level in this experiment.

Experiments 2 and 3 emphasize the aforementioned two points by injecting a disturbance followed by simultaneously targeting BPCS and SIS with a DoS attack and studying the reactor behavior under these attack conditions.

6.2. Experiment 2 - [Integrity → DoS] Attack causing a Reactor Overflow

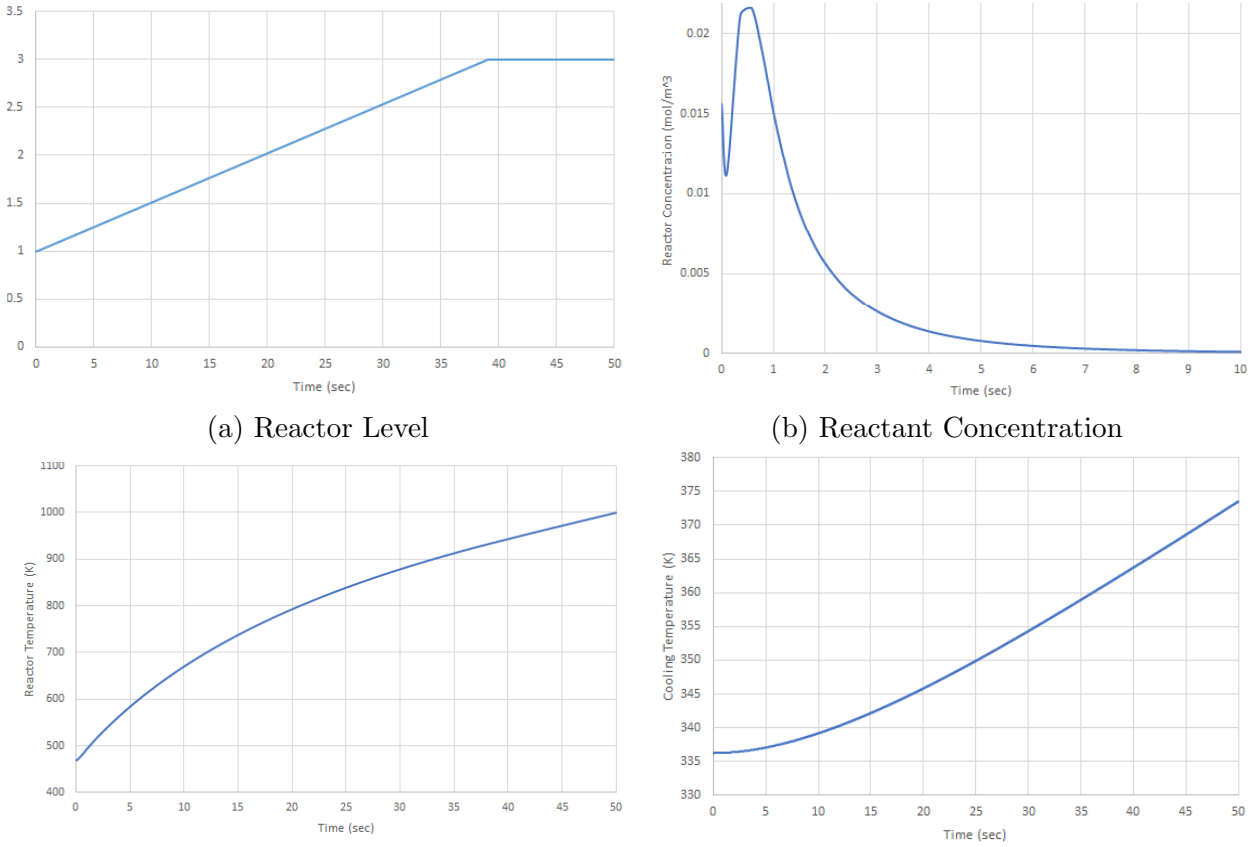
An integrity attack was launched from the HMI by fully closing the reactor outlet valve and fully opening the reactant inlet valve. Then, a DoS attack was launched against *both* the BPCS and SIS controllers from the compromised HMI workstation. The experiment prerequisites are the same as experiment 1. The conducted experiment was as follows:

- $t = 0$: HIC-03 controller was set to manual mode and the control valve CV-3 opening was set to 0%. LIC-01 controller was set to manual mode and the control valve CV-1 opening was set to 100%.
- $t = 0$: An ICMP flood DoS attack was launched from the compromised HMI against the BPCS and SIS simultaneously using hping3 penetration testing tool as in Experiment 1. For the SIS, the following command was executed

$$\text{hping3 -1 –flood –spoof} \quad \underbrace{192.168.0.2}_{\text{HMI}} \quad \underbrace{192.168.0.4}_{\text{SIS}}$$

Figure 16 presents the reactor state variables during the experiment.

- Reactant Level: As the process is out of control, and the SIS is lost as well, the level monotonically increased until the reactor overflows when the reactant level reaches 3 m at about $t = 39$ sec. The reactor overflow poses a safety hazard as per Table 2.
- Reactant Concentration: The concentration exhibited an oscillatory behavior for a very short period of time followed by an exponential decay to a near-zero concentration level. The drop of the reactant concentration is due to the rapid increase in the reactor temperature as explained below.
- Reactor Temperature: By increasing the inlet flow and reducing the outlet flow to zero, the state space model shows that the rate of change of the reactor temperature increases significantly, and is balanced out only by the heat exchange with the cooling fluid. Therefore, the reactor temperature increases rapidly where it reached the maximum operating limit of the reactor (800 K) in 22 sec.
- Cooling Temperature: Due to the significant increase in the reactor temperature, the cooling temperature increased exponentially as well due to the heat transfer.



(a) Reactor Level

(b) Reactant Concentration

(c) Reactor Temperature

(d) Cooling Temperature

Figure 16: Experiment 2: Integrity and BPCS / SIS DoS Attack - Reactor Overflow

More sophisticated planned attacks could cause an immediate process damage. The time span from the attack injection to the occurrence of the hazardous event depends mainly on the process dynamics. For the given reactor process, the process dynamics are fast enough such that a reactor hazard could be caused in less than 25 sec from the attack launch.

This experiment shows the importance of safeguarding against simultaneous BPCS and SIS attacks for the class of processes that do not (or can not) have mechanical protection against safety hazards. For such processes, the SIS is the last line of defense, and losing it causes an immediate hazard, leaving only mitigation layers to limit the hazard consequences.

Finally, industrial SIS usually has sophisticated diagnostic capabilities that may detect the inability of the SIS CPU to update field control signals, even if the SIS security measures were poorly implemented. Therefore, a successful ICMP attack as presented in this experiment main fail to halt the SIS and the SIS may be able to bring the process to a safe shutdown.

6.3. Experiment 3 - [Integrity → DoS] Attack causing a Reactor Runaway

An integrity attack was launched from the HMI by fully closing the reactor outlet valve and fully opening the reactant inlet valve. Then, a DoS attack was launched against *both*

the BPCS and SIS controllers from the compromised HMI workstation. The experiment prerequisites are the same as experiment 1. The conducted experiment was as follows:

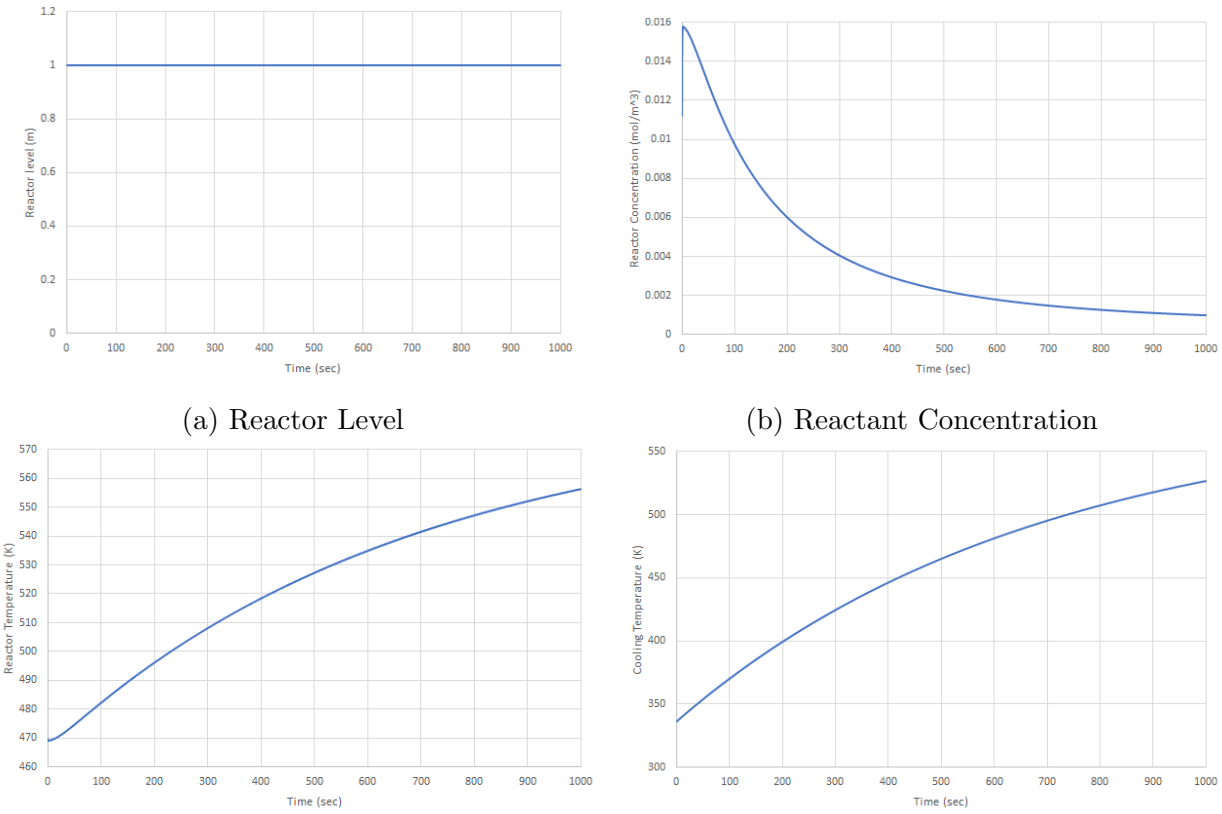
- $t = 0$: TIC-01 controller was set to manual mode and the control valve CV-2 opening was set to 0%.
- $t = 0$: An ICMP flood DoS attack was launched from the compromised HMI against the BPCS and SIS simultaneously as per Experiment 2.

For the following discussion, refer to Figure 17 for the reactor state variables during the experiment.

- Reactant Level: As there is no disturbance that impacts the reactant level, the level is stabilized throughout the experiment at its set point of 1 m.
- Reactant Concentration: The concentration undergoes an exponential decay to a near-zero concentration level. The drop of the reactant concentration is due to the rapid increase in the reactor temperature as explained below.
- Reactor Temperature: As the coolant flow is stopped, the heat exchange rate with the reactor jacket decreases considerably, and therefore, the reactor temperature increases rapidly with time. However, the reactor temperature never reached the maximum reactor design limit of 800 K. In the experiment, it took the reactor about 17 minutes to reach the temperature of 560 K.
- Cooling Temperature: The coolant temperature increases due to the stagnant flow in addition to the reaction heat transfer. Similarly, the jacket temperature did not reach its maximum design limit of 800 K but stabilized at around 600 K.

This experiment shows that not all attacks could cause a process safety hazard. Although the process was upset resulting in an out-of-spec product and a financial loss due to the down time, the process was never driven to an unsafe state. This is mainly governed by the process dynamics and the type of attack injected. Accordingly, we could say that the process has an inherent protection against specific types of attacks or faults. This fact becomes important when designing the security system, as different attacks would have different process consequences and therefore their countermeasures should be designed separately. This emphasizes the importance of the model-based design approach pursued in this paper, not only for experimental design, but also for the design of the associated security system.

This section lists some general observations drawn from the three experiments conducted. First, the ICMP flood attack does not have to be launched from the HMI. It could run from any machine using spoofed IP addressing techniques, and safe IP list on the BPCS will not stop the attack. Second, the attack experiments assumed that it was possible to install a malicious code on the HMI, which may not be possible. In such cases, modification of existing software tasks could achieve the same objective. Third, the ICMP attack is an illegitimate traffic that could be stopped by several defense techniques. However, modification



(a) Reactor Level

(b) Reactant Concentration

(c) Reactor Temperature

(d) Cooling Temperature

Figure 17: Experiment 3: Integrity and BPCS / SIS DoS Attack - Reactor Runaway

of legitimate traffic to act maliciously is much harder to detect and counterfeit. Finally, embedded controllers have limited resources. Attacks that are designed to bring down a powerful workstation or server in minutes could bring down an embedded controller in a matter of seconds.

7. Discussion

The current industry practice is to design the cyber security defense system for the CPS using classical IT approaches, such as firewalls, access control lists, and intrusion detection systems. Often times, the adoption of these approaches in CPS domain leads to equal treatment of all system nodes and functions. Cyber Physical Systems have a specific intended function, such as the reactor control and safety shutdown for the presented testbed. An optimal cyber security design, both in terms of cost-effectiveness and reliability in physical process protection, has to incorporate both the physical system knowledge and the system software architecture. Model-based design incorporates the physical process model, hardware and software architecture, communication architecture, and the associated cyber attack models in one unified framework to design an optimal security system.

In order to design a security system for CPS, the given CPS has to be thoroughly tested to reveal its weaknesses against cyber attacks. In this paper, we presented a model-based systematic approach to generate cyber attack scenarios that are relevant to the CPS under test. Figure 18 summarizes this approach, which represents the second step in the model-based design process presented in Figure 2.

Figure 18: Model-Based Approach to Generate CPS Attack Experiments

The testbed architecture presented in this paper is a standard architecture for industrial control systems promoted by NIST [10]. Therefore, results drawn from the experiments conducted on the testbed could be beneficial for other ICS systems. Furthermore, the testbed is built in sufficient details to experiment, formulate, and evaluate the model-based design approach. All testbed components have associated models that will be used during further research work. The testbed will be used to experiment with various cyber attacks, and the models presented here will be refined to provide an accurate representation of the actual CPS. The security system will then be designed using model-based approaches, and the design will be tested against actual attacks on the testbed. Figure 2 illustrates the model-based design approach we will pursue using the presented testbed.

8. Conclusion

This paper presents a model-based design approach for CPS security design. A complete description for a testbed built at Qatar University to research the model-based approach was presented. The testbed hardware and software architectural models were developed, and a systematic methodology to generate cyber attack experiments was explained.

The paper shows that the model-based approach has the power to generate the most relevant attack scenarios to the CPS under test. This results in cost and time savings during different stages of CPS design. In addition, it is anticipated that the model-based approach will lead to more effective countermeasures design.

As an extension for the testbed, two technologies will be adopted. First, virtualization will be used for engineering workstations, operator workstations, servers, and associated networking equipment. Virtualization is a new adopted trend in the process industry that has the advantage of centralized management and reduced expansion cost. Second, we will adopt the distributed approach where sensors and actuators are autonomous nodes communicating over a digital bus without a centralized controller. Several digital buses have already been adopted in the process industry including Foundation Fieldbus and Profibus.

The paper presents only the first two steps in our research outlook. As a future work, we will conduct thorough attack experiments and analyze the results. We will use data analysis results to design the security countermeasures. We will investigate how to correlate

the experimental results and the design of the countermeasures. As an ultimate objective of the research, we will explore the integration of the proposed methodologies and algorithms in existing design tool chains.

Acknowledgment

This research was made possible by NPRP 9-005-1-002 grant from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] C. Alcaraz, S. Zeadally, Critical infrastructure protection: Requirements and challenges for the 21st century, International Journal of Critical Infrastructure Protection 8 (2015) 53–66. doi:10.1016/j.ijcip.2014.12.002.

[2] A. Hahn, A. Ashok, S. Sridhar, M. Govindarasu, Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid, IEEE Transactions on Smart Grid 4 (2) (2013) 847–855. doi:10.1109/TSG.2012.2226919.

[3] S. Poudel, Z. Ni, N. Malla, Real-time cyber physical system testbed for power system security and control, International Journal of Electrical Power & Energy Systems 90 (2017) 124–133. doi:10.1016/j.ijepes.2017.01.016.

[4] J. Jarmakiewicz, K. Parobczak, K. Maślanka, Cybersecurity protection for power grid control infrastructures, International Journal of Critical Infrastructure Protection 18 (2017) 20–33. doi:10.1016/j.ijcip.2017.07.002.

[5] P. A. Craig, Metrics for the National SCADA Test Bed Program (October).

[6] T. Inl, Idaho National Laboratory, National SCADA Test Bed Substation Automation Evaluation Report, Tech. Rep. October (2009). doi:10.2172/968658.

[7] X. Koutsoukos, G. Karsai, A. Laszka, H. Neema, B. Potteiger, P. Volgyesi, Y. Vorobeychik, J. Sztipanovits, SURE: A Modeling and Simulation Integration Platform for Evaluation of SecUre and REsilient Cyber-Physical Systems (jan 2017). doi:10.1109/JPROC.2017.2731741.

[8] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, R. Reddi, A control system testbed to validate critical infrastructure protection concepts, International Journal of Critical Infrastructure Protection 4 (2) (2011) 88–103. arXiv:1303.7397, doi:10.1016/j.ijcip.2011.06.005.

[9] A. P. Mathur, N. O. Tippenhauer, SWaT: a water treatment testbed for research and training on ICS security, in: 2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), IEEE, 2016, pp. 31–36. doi:10.1109/CySWater.2016.7469060.

[10] Stouffer, Keith, Joe Falco, K. Scarfone., Guide to industrial control systems (ICS) security., Tech. rep., NIST (2015). doi:800.82.

[11] T. E. Marlin, Process Control: Designing Processes and Control Systems for Dynamic Performance, 2000.

[12] ISA, ANSI/ISA-5.1-1992, Instrumentation Symbols and Identification, Tech. rep. (1992).

[13] IEC, IEC 61511-1:2016, Functional safety - Safety instrumented systems for the process industry sector - Part 1: Framework, definitions, system, hardware and application programming requirements, Tech. rep., IEC (2016).

[14] National Instruments LabVIEW Real-Time Module.
URL `http://www.ni.com/labview/realtime/`

[15] LabVIEW Datalogging and Supervisory Control (DSC) Module - National Instruments.
URL `http://www.ni.com/labview/labviewdsc/`

[16] W. Stallings, L. Brown, Computer Security : principles and practice (4th edition), Pearson, 2018.