NISTIR 8177

# Metrics and Key Performance Indicators for Robotic Cybersecurity Performance Analysis

Timothy A. Zimmerman

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8177

National Institute of
Standards and Technology
U.S. Department of Commerce

## NISTIR 8177

# Metrics and Key Performance Indicators for Robotic Cybersecurity Performance Analysis

Timothy A. Zimmerman
*Intelligent Systems Division*
*Engineering Laboratory*

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8177

April 2017

U.S. Department of Commerce
*Wilbur L. Ross, Jr., Secretary*

National Institute of Standards and Technology
*Kent Rochford, Acting NIST Director and Under Secretary of Commerce for Standards and Technology*

# CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# 1 INTRODUCTION

The National Institute of Standards and Technology has constructed a testbed to measure the performance impact induced by cybersecurity technologies on Industrial Control Systems (ICS) [1]. The testbed allows researchers to emulate real-world industrial manufacturing processes and their control systems without the need to replicate an entire factory environment or its machinery.

The focus of this report is the Robotic Enclave of the testbed, which is comprised of two robotic arms that emulate a material handling application, known as “machine tending.” Robotic machine tending uses robots to interact with the machinery, performing operations a human operator would normally perform (e.g., the loading and unloading of parts, opening and closing of machine doors, activating operator control panel buttons). In the enclave, parts are transported collaboratively through simulated sequential machining operations, known as “stations.” The enclave was designed and constructed to be reconfigurable, allowing numerous types of operational methodologies, network topologies, and industrial networking protocols to be investigated.

The research performed on the enclave will explore the effects of implementing cybersecurity technologies, as defined by industry best-practices and standards, on the testbed operation and measure the performance impact. As later sections will elaborate, the impact will be measured simultaneously across many of the enclave subsystems during an experiment to provide a holistic understanding of the underlying effects of the defenses.

This report identifies the metrics, Key Performance Indicators (KPI), their derivations, and the measurement methodologies that will be employed during future enclave experiments.

# 2 ROBOTIC ENCLAVE

The Robotic Enclave, shown in Figure 1, is one of four enclaves within the Cybersecurity for Smart Manufacturing Systems (CSMS) testbed. It includes two robotic arms that emulate a machine tending application, where parts are loaded, unloaded, and transported between sequential machining operations in a simulated batch-production process.

**Figure 1 - View of the robotic enclave and its components**

The robots operate in concert according to a material handling procedure that changes dynamically based on feedback from the simulated machining operations. In addition to the two industrial robots, the enclave includes a supervisory programmable logic controller (PLC), a safety PLC, a human machine interface (HMI), a real-time vision tracking system, several servers for executing required computational resources and applications, and an engineering workstation.

## 2.1 SERVER OPERATIONS

The robot controllers can operate in one of two modes: deployed or virtualized. In the deployed mode, each robot is controlled on a dedicated Dell PowerEdge R420 server running the Robot Operating System (ROS) on top of Ubuntu Linux. In the virtualized mode, each robot is controlled by virtualized servers within a hypervisor running on a Dell PowerEdge 620 server. The deployed mode supports experiments with a pseudo-ideal configuration, and the virtualized mode supports experiments with a resource-restricted configuration, as well as the ability to maintain independent testing environments.

The pseudo-ideal configuration provides the robot controller software with computational resources well-beyond the minimum requirements for unimpeded operations. Operating in this manner is reserved for experiments that do not require server performance impacts to be measured (e.g., network-specific experiments). The resource-restricted configuration allows the researchers to restrict the available resources to the robot controller software and underlying operating system (e.g., memory allocation, available hard disk space, hard disk access rates, number of central processing unit (CPU) cores).

The hypervisor also allows software-based cybersecurity tools to be deployed within an isolated environment for testing and the ability to restore the enclave environment to a known-good state, reducing the chances of cross-contamination by residual software modules or services remaining within a virtual machine post-experiment. Software-based cybersecurity tools are installed on virtual machines dedicated to specific experiments within the hypervisor, and archived. This allows any tool to be recalled for any experiment that requires its execution.

## 2.2 EMULATED MANUFACTURING OPERATION

The sequential machining operations emulated by the enclave are shown in Figure 2. The robots work collaboratively to move the parts between the successive stations. Each station is physically constructed from an additive manufactured base with a concave recess to capture the parts, and an integrated proximity sensor. The parts transported between the stations are acetal resin spheres, 38 mm (1.5 inch) in diameter.

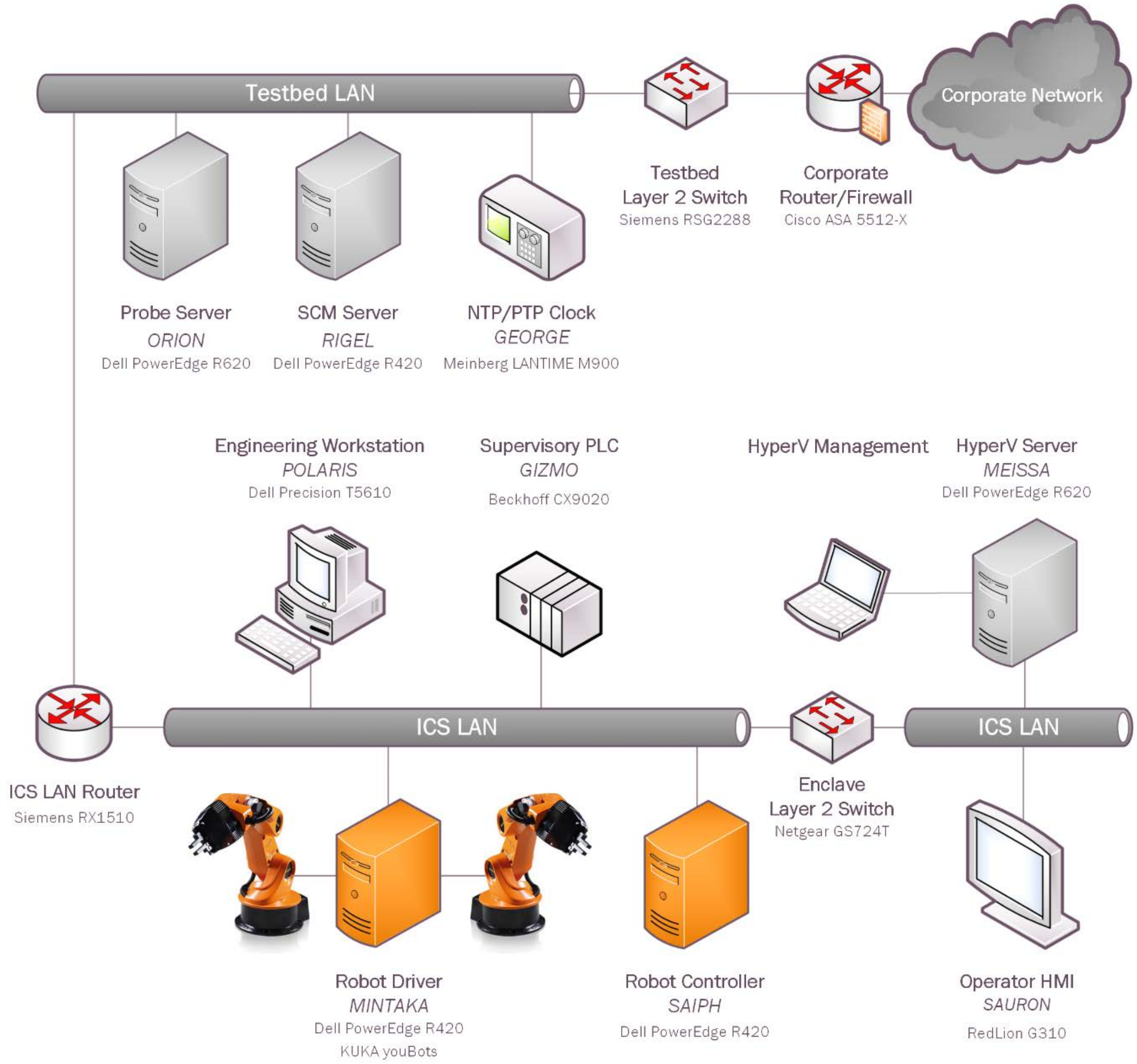
**Figure 2 - Flow chart of the emulated manufacturing process**

The machining operations of each station are emulated as configurable timers within the supervisory PLC (Beckhoff CX9020) with three operational states: “idle,” “processing,” and “ready for pickup.” A station is in the “idle” state when no part is present, the “processing” state when a part is present and the timer has not reached its set duration, and the “ready for pickup” state when a part is present and the timer has reached its set duration. The status information for each machining station is made available to other network devices (e.g., robot controllers, HMI) via a Modbus TCP server on the PLC.

## 2.3 NETWORK ARCHITECTURE

The enclave local area network (ICS LAN) is constructed as a flat architecture, as shown in Figure 3. For the initial experimentation, the architecture will remain flat (as this is most prevalent in industry). However, the reconfigurable design of the enclave will enable the implementation of network segmentation and security perimeters during future experiments. The local network traffic ("ICS LAN") is managed by a Siemens RUGGEDCOM RX1510, and the high-level testbed traffic ("Testbed LAN") and its connection to the "Corporate Network" are managed by a Cisco ASA 5512-X.

**Figure 3 - Diagram of the network architecture and its components**



The Testbed LAN has three machines: a probe server, source control management (SCM) server, and Network Time Protocol (NTP) and Precision Time Protocol (PTP) capable master clock. The probe server ("ORION") consumes and records network traffic from twelve independent network interfaces. The SCM server ("RIGEL") runs a GitLab server to track and maintain the code used on the enclave, primarily robot code, PLC code, and analysis tools. The NTP/PTP clock ("GEORGE") provides stable time to the enclave servers and systems to enable synchronized logging.

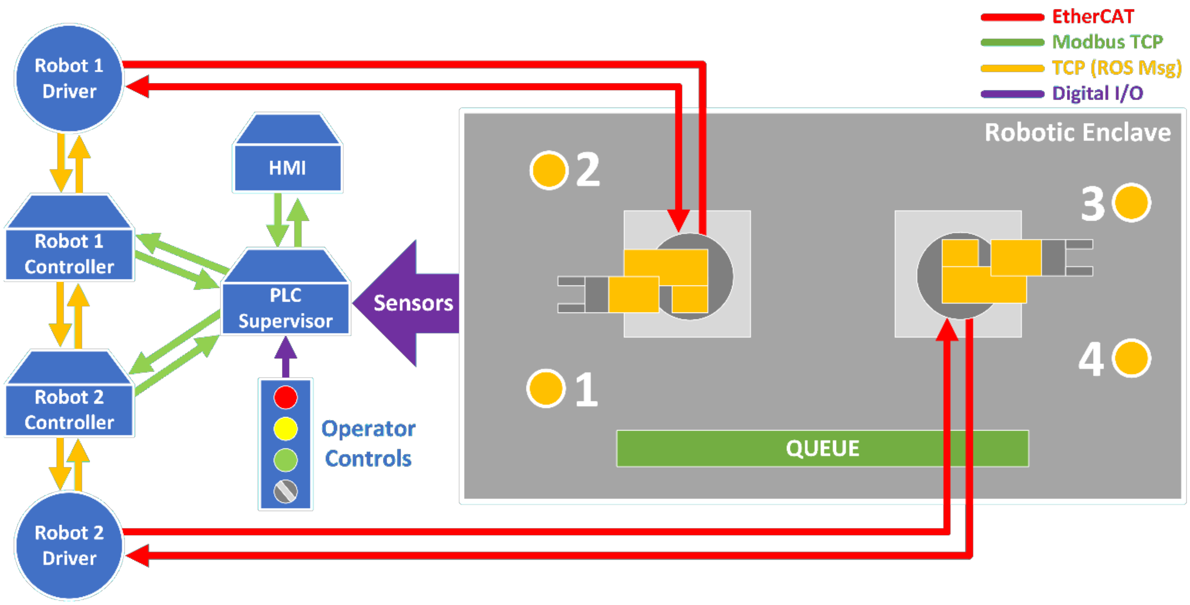
The ICS LAN has numerous machines that directly operate and support the operation of the enclave. The robot controller/driver servers (“MINTAKA”, “SAIPH”) execute the operational code and communicate directly with the robots to direct their actions. The supervisory PLC (“GIZMO”) communicates the status of the machining stations and operator controls to the robot controllers, as well as part tracking for manufacturing performance measurements. The operator HMI also communicates with the PLC to display manufacturing process information and performance measurements to the operator. The engineering workstation (“POLARIS”) hosts the programming environment and debugging tools used to modify the robot code, and give terminal-level access to other machines within the enclave. The HyperV server (“MEISSA”) provides server virtualization to the enclave, allowing researchers to create servers on-demand, as required by specific software tools or packages.

Network probes within the enclave intercept network traffic at key points and mirror the traffic to the measurement enclave via a patch panel. Within the measurement enclave, packets can be captured, manipulated, and logged using a collection of network testing hardware and software.

## 2.4 PROCESS OPERATIONS AND DATA FLOW

Figure 4 illustrates the data flow of the processes and components within the enclave. All operations are initiated by the PLC, which monitors the operator buttons, HMI, and safety systems. The robots will commence their material handling operations only when the PLC informs the robot controllers that it is safe to do so. This communication is enabled by a Modbus TCP server in the PLC, which the robot controllers can access directly via Modbus TCP polling. The Modbus server provides the controllers with the current operating state of the enclave, the status of each operator button and switch, and the status of each machining station.

**Figure 4 - Data flow diagram**



While a robot is idle, the controller will continue polling the Modbus Service to obtain the operating status of each machining station. This status information is passed through decision logic within the robot controller to determine what actions the robot should perform, based on the given machining station states. Once the controller logic determines which action should be executed, the arm will move through a series of predetermined poses and actions to complete the intended operation.

The predetermined poses and actions are described in a Yet Another Markup Language (YAML) file, which defines each discrete joint position or action (e.g., open/close the jaw) the arm should sequentially execute to complete the movement. For example, if Station #1 reports a part is ready for pickup, Robot #1's controller will check to see if there is a part at Station #2. If Station #2 is empty, the robot controller will execute the required set of actions to move the part from Station #1 to Station #2. The controller will continue executing each discrete action within the set until it has completed. At this point, the controller will return the robot to its idle position and continue scanning the machining stations until a new event triggers another movement to be performed (e.g., move a part from the raw material queue to the now empty Station #1).

Parts will continue moving through the machining stations as long as there is raw material on the queue, and the experiment termination triggers within the HMI are not achieved. Experiment termination triggers available can be either part-counter based, or timer-based. Once a trigger is activated, the enclave will enter the "stop" state and allow the robots to complete their current movements before ceasing all operations.

The control of each robot joint is performed with a cascaded control architecture. Closed-loop proportional-integral-derivative (PID) position control is performed by the motor controllers for each joint. The enclave also has the capability to control the robot joints in a velocity control mode, where the PID position control is performed by the robot controllers, and velocity updates are sent over the network.

Because of the reconfigurable nature of the enclave, the hosts of the robot controller processes are excluded from Figure 4. However, in the deployed mode, the robot controller processes typically execute on SAIPH, with the driver processes executing on MINTAKA.

# 3 METRICS AND KPI

The research and selection of enclave metrics and KPI were driven by the experimental goal: to capture empirical evidence of performance impacts induced by cybersecurity protections (e.g., hardware, software, implementation of industry best-practices). Measurements of the manufacturing operation are captured and compared to those captured during baseline experiments. Statistical analysis will provide a method of detecting deviations from the expected behavior of the manufacturing process, although it is not the focus of this report.

The deviation of a specific measurement from the baseline will not directly identify the underlying cause. To resolve this, measurements from the subsystems that support the manufacturing operation are also captured. Simultaneously measuring at the manufacturing-level and subsystem-level enables analysis of the performance impact to the manufacturing operation, as well as the underlying subsystems and their interactions. In the robotic enclave, the high-level system is the discrete manufacturing operation, and supporting the operation are the enclave subsystems (e.g., servers, PLC, HMI, network equipment).

For an example of the interactions between the manufacturing process and underlying subsystems, consider a hypothetical manufacturing operation where an operator must regularly inspect the output product and manipulate machine variables via an HMI to keep the parts free of defects. The proposed cybersecurity protection to be implemented is "requiring a password on the HMI to prevent unauthorized users from manipulating the machine variables." If the machine in question produces products at a rate of 10 parts per hour, and it takes the operator an extra 5 seconds to unlock the HMI, the performance impact to the manufacturing process will be negligible. However, if the machine operates at high-speeds, with production rates measured in parts per second, requiring the operator to regularly spend an additional 5 seconds to unlock the HMI and adjust the machine variables will cause performance impacts to the output production rate.

## 3.1 TERMINOLOGY

This report uses a naming convention that follows terminology to enable effective communications between researchers and industry, as proposed in [2]. More specifically, this report defines a metric as a "directly measurable property of a system," while a KPI is defined as a "computable performance assessment, as derived from a combination of metrics." Metrics are analogous to primitives, while KPIs are a form of post-analysis which utilize the primitives to compute an inference. For comparison, ISO 22400-1:2014 defines the measurements used in the calculation of KPIs as "elements."

Using the robotic enclave as an example, consider the following two metrics: the amount of time a single machining station is performing work on parts during a batch, and the total amount of time required to produce the batch. Both are directly measurable, and are considered metrics. However, effective communication of either metric proves problematic when comparing to systems with differing dynamics (i.e., an enclave with four machining stations compared to six machining stations). This is resolved by combining individual metrics within a computation to calculate a KPI. Continuing with the previous example, the machining station working time and total batch production time can be used to calculate the "Utilization Efficiency" of the station (see KPI 2.4 in Section 4.1), which can be directly communicated and compared to other systems.

## 3.2 SELECTION PROCESS

The manufacturing process and various subsystems operating within the enclave were examined to determine which metrics could be obtained. Literature related to the performance of manufacturing systems and other industry standards were then reviewed to determine which KPI could be used to best analyze the manufacturing process with the available metrics. The literature in [3], [4], [5], [6], and [7] proved most helpful by providing insight into some of the more elusive KPI for the testbed subsystems. KPIs that did not provide results related to the research goal were not considered.

# 4 MEASUREMENT METHODOLOGIES

To detect and measure performance impacts to the manufacturing system, specific metrics from the manufacturing process, robots, local network, PLC, and servers are captured. These metrics are then used to calculate KPIs to provide quantifiable and communicable indicators, and detect any performance impacts. If any impacts are detected, more detailed analysis can be performed to assist with locating the source of the performance degradation.

The following sections describe the measurement methodology used for each process and subsystem, as well a summary of the metrics and KPIs. Detailed descriptions and formulas for each metric and KPI are located in Appendix A – Description of Measurements and KPI.

## 4.1 MANUFACTURING PROCESS PERFORMANCE

For each batch produced, the PLC captures all the metrics listed in Table 1. Arrival and departure timestamps for each station are stored in a multidimensional array, and the station counters are implemented as primitive variables. When the production process is started, the PLC will log the start time obtained from the grandmaster NTP clock (labeled GEORGE in Figure 3). Once the experiment start time is logged, the PLC enables enclave operations: Modbus server registers are updated to inform all subsystems that operations are enabled, the counters begin incrementing, and the recording of each station event is enabled.

Counters and event trackers are updated on every cycle of the PLC task, which is 10 ms at the time of writing. When a part enters or exits a station, a timestamp of the event is stored in the part tracking array, relative to the start of the batch. The PLC also captures the total amount of time each station spends in any valid state. The three valid states for a machining station are: busy, waiting, and idle. The busy state represents the time a station is actively machining a part. After machining has completed, the station enters the waiting state, and waits for a robot to remove the part. Once the part is removed from the station, it enters the idle state until a new part is loaded. Batch production time and the total number of parts produced are also tracked, as they are required for some of the KPI calculations.

When the enclave completes its production cycle, all process data remains within the PLC until it is downloaded and purged by an operator via the HMI. The KPIs, listed in Table 2, can be computed in real-time via the operator HMI, or offline after the experiment has been completed.

**Table 1 - Manufacturing Process Metrics**

| Metric | Description |
|---|---|
| 1.1 Part Timestamps | Timestamping of unique part events (e.g., arriving and departing stations) |
| 1.2 Station Timers | Accumulation of time a station is in any valid state |
| 1.3 Batch Timer | Total amount of time required for the batch to be produced |
| 1.4 Batch Parts Counter | Number of parts produced in the batch |

**Table 2 - Manufacturing Process KPI**

| Key Performance Indicator | Description |
|---|---|
| 2.1 Part Production Time | Amount of time required for a part to be produced |
| 2.2 Cycle Time | Amount of time between finished parts |
| 2.3 Throughput Rate | Number of parts produced over a specific amount of time |
| 2.4 Production Effectiveness | Relationship between the planned production time to complete a batch and the actual production time |
| 2.5 Station Allocation Ratio | Ratio between the amount of time a station was busy and the batch production time |
| 2.6 Station Utilization Efficiency | Ratio between the amount of time a station was producing and the amount of time the batch required to complete |

## 4.2 ROBOT PERFORMANCE

Robot performance metrics can be obtained by capturing the command and control communications used by the robots. Since the communication protocol used by the robots (EtherCAT) leverages existing Ethernet hardware, the packets can be captured directly from the physical interface, or with an in-line tool. Each packet contains all of the metrics shown in Table 3, and are transmitted at a rate of 700 Hz. This communication is the lowest-level that can be captured, as packets on this path go directly to each individual joint of the robots.

Because of the huge amount of data being transmitted during a batch, capturing data on this interface is time-limited, and is typically reserved for experiments where robot performance impact is anticipated because of interim results obtained from more readily available metrics and KPIs.

When a capture of this interface is performed, all of the network traffic is logged through a Wireshark or `tcpdump` capture. The EtherCAT protocol data is then extracted from the packets offline by a custom Python dissector script.

**Table 3 - Robot Performance Metrics**

| Metric | Description |
|---|---|
| 3.1 Joint Position | Position of each robot joint |
| 3.2 Joint Setpoint | Control setpoint of each robot joint |
| 3.3 Joint Velocity | Velocity of each robot joint |
| 3.4 Joint Current | Electrical current of each robot joint |
| 3.5 Joint Control Flags | Flags describing the operational status of each robot joint |

**Table 4 - Robot Performance KPI**

| Key Performance Indicator | Description |
|---|---|
| 4.1 Actuation Latency | Amount of time for the robot arm to move after a command is sent |
| 4.2 Pose Travel Time | Amount of time for the robot arm to move between two poses |
| 4.3 Position Accuracy | Deviation between the commanded position and the measured position |
| 4.4 Position Repeatability | Closeness of agreement between the measured position after *n* repeat visits to the same commanded position |
| 4.5 Energy Consumption | Estimation of the amount of power consumed by the robot joint motors |

## 4.3 NETWORK PERFORMANCE

The network capture server within the measurement enclave is tasked with capturing all transmitted packets within the local network. Each packet is stored in its raw format, allowing numerous metrics to be obtained. Key nodes within the enclave have their Ethernet communications passed through an in-line Ethernet network tap (USRobotics USR4503). The packets are aggregated and mirrored to the measurement rack, where they are captured by a Dell PowerEdge R620 containing two Broadcom 5719 network interface cards, adding eight dedicated Ethernet packet capture interfaces.

A reconfigurable Python script and dissector were created to perform the capture and dissection processes. The script creates a dedicated `tcpdump` process for each Ethernet interface, as defined by the user in a configuration file, and stores captured packets in PCAP files. Dissection of ROS packets is performed using a Python script that contains the structures and signatures of known message types. The script can also automatically cease its measurements by listening for broadcast messages on the local network marking the end of an experiment.

**Table 5 - Network Performance Metrics**

| Metric | Description |
|---|---|
| 5.1 Packet Headers | Header data of a captured packet |
| 5.2 Packet Data | Application layer payload of a captured packet |
| 5.3 Packet Protocol Type | Protocol type of a captured packet |
| 5.4 Packet Timestamp | Timestamp of a captured packet |
| 5.5 Packet Size | Size of a packet in bytes |
| 5.6 Packet Errors | Number of packets transmitted or received with errors |
| 5.7 Dropped Packets | Number of packets dropped by the network |
| 5.8 Packet Counter | Number of packets transmitted and received |

**Table 6 - Network Performance KPI**

| Key Performance Indicator | Description |
|---|---|
| 6.1 Packet Path Delay | Time delay along the path from transmitter to receiver |
| 6.2 Inter-packet Delay | Difference between the packet path delay of two packets |
| 6.3 TCP Packet Round Trip Time | Amount of time for the source node of a packet to receive the acknowledgement of receipt (ACK) from the destination node |
| 6.4 Information Ratio | Ratio of the quantity of process information packets to all packets |
| 6.5 Bit Rate | Rate of bits transmitted or received over a specific timespan |
| 6.6 Packet Rate | Rate of packets transmitted and received over a specific amount of time |
| 6.7 Packet Error Rate | Rate of packets transmitted or received with errors over a specific amount of time |
| 6.8 Proportion of Protocol Type | Numerical proportion of a unique packet protocol type observed |

## 4.4 PROGRAMMABLE LOGIC CONTROLLER PERFORMANCE

The PLC simulating the machining stations and communications with the robot controllers requires CPU resources to perform its required functions. Any performance impacts to the PLC operations will be evident through the recorded metrics described below in Table 7. In contrast to the PLC's responsibilities described in Section 2.4, the KPIs in this section are related to the operational performance of the PLC itself, whereas the KPIs in Section 4.1 are related to the operational performance of the manufacturing process.

The TwinCAT TC3 Engineering module (which executes on the PLC) includes two library functions for capturing the metrics listed in Table 7: `TC_CpuUsage`, and `FB_CxProfiler`. The metrics from these two functions are recorded once every second while the experiment is running. All data is stored within the PLC until cleared by an operator via the HMI after the data has been downloaded.

**Table 7 - PLC Performance Metrics**

| Metric | Description |
|---|---|
| 7.1 Task Execution Time | Time required to complete the PLC task |
| 7.2 Minimum Task Execution Time | Minimum amount of time required to complete the PLC task |
| 7.3 Maximum Task Execution Time | Maximum amount of time required to complete the PLC task |
| 7.4 CPU Utilization | Amount of PLC CPU usage |
| 7.5 Maximum CPU Utilization | Maximum amount of PLC CPU usage |

**Table 8 - PLC Performance KPI**

| Key Performance Indicator | Description |
|---|---|
| 8.1 Task Execution Time | Mean and standard deviation of PLC task execution time |
| 8.2 CPU Utilization | Mean and standard deviation of PLC CPU utilization |

## 4.5 SERVER PERFORMANCE

The servers within the enclave require computing resources (e.g., CPU time, memory usage, disk usage) to perform their tasks during an experiment (e.g., robot control). To capture the performance of these resources, a Python application was created to capture and log the required metrics. All of the metrics are logged in comma-separated values files at a specified rate, which is defined by the researcher through command line arguments (the default rate is one sample per second).

To perform the capture, the researcher executes the script from the command line interface, along with required arguments, before the experiment is initiated. Once initiated, the script will continue capturing the metrics until it is stopped. The script can also automatically cease its measurements by listening for broadcast messages on the local network that indicate the end of an experiment.

Although there is a separate measurement system capturing the network traffic of the enclave, it is important to note the inclusion of server-based network measurements (Metrics 9.7-9.10). Some of the cybersecurity tools used during experiments block traffic from the wire, before a packet has the chance to reach a server. Measuring at the server allows a comparison of traffic flow on the network to what is actually consumed and generated by a specific server. However, one important difference between the network capture and server performance measurements is the latter does not include raw packet data; only simple flow counters.

**Table 9 - Server Performance Metrics**

| Metric | Description |
|---|---|
| 9.1 CPU Utilization Timers | Amount of server CPU utilization |
| 9.2 Available Memory | Amount of server memory available by the server |
| 9.3 Total Memory | Total amount of memory installed in the server |
| 9.4 Disk I/O Byte Counters | Number of bytes read and written to the server hard drive(s) |
| 9.5 Disk I/O Access Counters | Number of discrete read and write operations to the server hard drive(s) since the last sample |
| 9.6 Disk I/O Time | Amount of time used to read and write to the server hard drive(s) since the last sample |
| 9.7 Network Byte Counters | Number of bytes transmitted and received by the server |
| 9.8 Network Packet Counters | Number of packets transmitted and received by the server |
| 9.9 Network Packet Errors | Number of packets with errors transmitted and received by the server |
| 9.10 Network Dropped Packets | Number of incoming or outgoing packets dropped by the server |

**Table 10 - Server Performance KPI**

| Key Performance Indicator | Description |
|---|---|
| 10.1 CPU Utilization | Mean and standard deviation of server CPU utilization |
| 10.2 Memory Utilization | Mean and standard deviation of server memory utilization |
| 10.3 Average Disk I/O | Mean rate and standard deviation of data read and written to the server hard drive(s) |
| 10.4 Network Throughput (Bits) | Mean rate and standard deviation of bits transmitted and received by the server |
| 10.5 Network Throughput (Packets) | Mean rate and standard deviation of packets transmitted and received by the server |

## 5 APPENDIX A – DESCRIPTION OF MEASUREMENTS AND KPI

| | |
|---|---|
| **ID** | 1.1 |
| **Name** | Part Timestamps ($p_A^{x,S}$, $p_D^{x,S}$) |
| **Description** | Timestamping of unique part events (e.g. arriving and departing stations) |
| **Unit of measure** | seconds |
| **Notes** | Each part has an arrival and departure timestamp for each station, where $p_A^{x,S}$= the arrival timestamp of part $x$ as station $S$, and $p_D^{x,S}$ is the departure of timestamp of part $x$ as station $S$.<br>When a part arrives or departs a simulated machining station, the PLC will log the time. The PLC tracks time as discrete 10 millisecond time steps from the start of the production cycle. The timestamps are converted to seconds for analysis. |

| | |
|---|---|
| **ID** | 1.2 |
| **Name** | Station Timers ($s_P^S, s_F^S, s_I^S$) |
| **Description** | Accumulation of time a station is in any valid state |
| **Unit of measure** | seconds |
| **Notes** | Simulated machining stations have three operating states:<br>• Processing – station is loaded and the part is being processed<br>• Finished – station is loaded and waiting for the robot to remove the part<br>• Idle – station is unloaded and waiting for a part to be delivered |

| | |
|---|---|
| **ID** | 1.3 |
| **Name** | Batch Timer ($b_P$) |
| **Description** | Total amount of time required for the batch to be produced |
| **Unit of measure** | seconds |
| **Notes** | The timer is started after the operator puts the enclave into the operational state, and is stopped when enclave operations end (automatic or operator shutdown). |

| | |
|---|---|
| **ID** | 1.4 |
| **Name** | Batch Parts Counter ($b_Q$) |
| **Description** | Number of parts produced in the batch |
| **Unit of measure** | quantity |

| | |
|---|---|
| **ID** | 2.1 |
| **Name** | Part Production Time ($E_P$) |
| **Description** | Amount of time required for a part to be produced |
| **Formula** | $$E_P^x = p_D^{x,S_n} - p_A^{x,S_0} \quad ,$$<br>where $E_P^x$= production time of part $x$; $p_D^{x,S_n}$= departure timestamp of part $x$ at the last station, $s_n$; and $p_A^{x,S_0}$= arrival timestamp of part $x$ at the first station, $s_0$. |
| **Unit of measure** | seconds |
| **Notes** | This metric is calculated by the PLC after the part has departed the final station. |

| | |
|---|---|
| **ID** | 2.2 |
| **Name** | Cycle Time ($E_C$) |
| **Description** | Amount of time between finished parts |
| **Formula** | $$E_C^x = p_D^{x,S_4} - p_D^{x-1,S_4} \quad ,$$<br>where $E_C^x$= cycle time of part $x$; $p_D^{x,S_4}$= departure timestamp of part $x$ at station $s_4$; and $p_D^{x-1,S_4}$= departure timestamp of the previous part, $x-1$, at station $s_4$. |
| **Unit of measure** | seconds |

| | |
|---|---|
| **ID** | 2.3 |
| **Name** | Throughput Rate ($E_R$) |
| **Description** | Number of parts produced over a specific amount of time |
| **Formula** | $E_R^T = \frac{\Delta b_Q^T}{t_i - t_0}$<br>with<br>$T = t_0 \leq t < t_i \quad ,$<br>where $E_R^T$= throughput rate during timespan $T$; and $\Delta b_Q^T$= change in the batch part counter during timespan $T$. To calculate the throughput of the total batch, the formula can be rewritten as:<br>$E_R = \frac{b_Q}{b_P} \quad ,$<br>where $E_R$= batch throughput rate; $b_Q$= quantity of parts produced in the batch; and $b_P$= total amount of time required to produce the batch |
| **Unit of measure** | parts / hour |
| **Notes** | Testbed metrics are recorded in seconds, requiring the formula to be rewritten as:<br>$E_R^T = \frac{\Delta b_Q^T}{t_i - t_0} * 3600^{-1}$<br>and<br>$E_T = \frac{b_Q}{b_P} * 3600^{-1}.$ |

| | |
|---|---|
| **ID** | 2.4 |
| **Name** | Production Effectiveness ($E_E$) |
| **Description** | Relationship between the planned production time to complete a batch and the actual production time |
| **Formula** | $E_E = \frac{\widehat{E_C} * b_Q}{b_P}$ ,<br>where $\widehat{E_P}$= estimated cycle time between finished parts; $b_Q$= the total number of parts produced in the batch; and $b_P$= total amount of time required to produce the batch. |
| **Unit of measure** | % |
| **Notes** | For the purpose of this experiment, the estimated part production time ($\widehat{E_P}$) is calculated by finding the mean cycle time between parts from the baseline experiments ($\overline{E_{Pb}}$):<br>$\widehat{E_C} = \overline{E_{Pb}} = \frac{1}{b_{Qb}} \sum_{x=1}^{b_{Qb}} E_{Cb}^{x}$ ,<br>where $b_{Qb}$ = number of parts produced during the baseline experiments, and $E_{Cb}^{x}$= cycle time between parts required to produce part $x$ during the baseline experiments. |

| | |
|---|---|
| **ID** | 2.5 |
| **Name** | Station Allocation Ratio ($E_A$) |
| **Description** | Ratio between the amount of time a station was busy and the batch production time |
| **Formula** | $E_A^S = \frac{s_P^S + s_F^S}{b_P}$ ,<br>where $E_A^S$= allocation ratio of station $S$; $s_P^S + s_F^S$= amount of time station $S$ was busy; and $b_P$= total amount of time required to produce the batch. |
| **Unit of measure** | % |
| **Notes** | A “busy” station is defined as any station that is not in the “idle” state. |

| | |
|---|---|
| **ID** | 2.6 |
| **Name** | Station Utilization Efficiency ($E_U$) |
| **Description** | Ratio between the amount of time a station was producing and the amount of time the batch required to complete |
| **Formula** | $$E_U^S = \frac{s_P^S}{b_P} \quad ,$$ where $E_U^S$= utilization efficiency of station $S$; $s_P^S$= amount of time station $S$ was in the “processing” state; and $b_P$= total amount of time required to produce the batch. |
| **Unit of measure** | % |

<table>
<tr><td>**ID**</td><td colspan="6">3.1</td></tr>
<tr><td>**Name**</td><td colspan="6">Joint Position ($j_{\theta P}$)</td></tr>
<tr><td>**Description**</td><td colspan="6">Position of each robot joint</td></tr>
<tr><td>**Unit of measure**</td><td colspan="6">radians</td></tr>
<tr><td rowspan="3">**Range**</td><td></td><td>**Joint 1**</td><td>**Joint 2**</td><td>**Joint 3**</td><td>**Joint 4**</td><td>**Joint 5**</td></tr>
<tr><td>**Minimum**</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>**Maximum**</td><td>5.90</td><td>2.70</td><td>5.16</td><td>3.60</td><td>5.87</td></tr>
<tr><td>**Notes**</td><td colspan="6">Joint encoder values are measured relative to the home pose values. The formula used to convert the robot encoder values to radians is:<br>$$j_\theta^h = \frac{|\eta_m^h|}{\eta_p} * \lambda^h * 2\pi \quad ,$$<br>where $j_\theta^h$= angle in radians of joint $h$; $\eta_m^h$= encoder value of joint $h$; $\eta_p$= rated encoder pulses per revolution; and $\lambda^h$= gear ratio of joint $h$.</td></tr>
</table>

| | | |
|---|---|---|
| **ID** | 3.2 | |
| **Name** | Joint Setpoint ($j_{\theta S}$) | |
| **Description** | Control setpoint of each robot joint | |
| **Unit of measure** | radians | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 5.90 |
| **Notes** | Joint encoder values are measured relative to the home pose values. The formula used to convert the robot encoder values to radians is:<br>$j_{\theta}^{h} = \frac{\lvert\eta_{m}^{h}\rvert}{\eta_{p}} * \lambda^{h} * 2\pi \quad ,$<br>where $j_{\theta}^{h}$= angle in radians of joint $h$; $\eta_{m}^{h}$= encoder value of joint $h$; $\eta_{p}$= rated encoder pulses per revolution; and $\lambda^{h}$= gear ratio of joint $h$. | |

| | |
|---|---|
| **ID** | 3.3 |
| **Name** | Joint Velocity ($j_{\dot{\theta}}$) |
| **Description** | Velocity of each robot joint |
| **Unit of measure** | radians per second |
| **Notes** | The formula used to convert the robot velocity data to radians per second is:<br>$j_{\dot{\theta}}^{h} = \frac{\omega^{h}}{60} * \lambda^{h} * 2\pi \quad ,$<br>where $j_{\dot{\theta}}^{h}$= velocity of joint $h$; $\omega^{h}$= revolutions per minute of the motor at joint $h$; and $\lambda^{h}$= the gear ratio of the transmission of joint $h$. |

| | |
|---|---|
| **ID** | 3.4 |
| **Name** | Joint Current ($j_I$) |
| **Description** | Electrical current of each robot joint |
| **Unit of measure** | amperes |
| **Notes** | Electrical current measurements are received from the robot in milliamperes. |

| | |
|---|---|
| **ID** | 3.5 |
| **Name** | Joint Control Flags ($j_F$) |
| **Description** | Flags describing the operational status of each robot joint |
| **Notes** | Flags are included for each joint, and include: communication timeout, initialization complete, position reached, joint controller mode, sensor error, motor halted, over-temperature, over-voltage, under-voltage, and over-current. |

| | |
|---|---|
| **ID** | 4.1 |
| **Name** | Actuation Latency ($R_L$) |
| **Description** | Amount of time required for the robot arm to begin movement after a command is sent |
| **Formula** | $R_L^P = r_a^P - r_c^P$ ,<br>where $R_L^P$= the actuation latency at position $P$; $r_a^P$= timestamp of actuation initiation at pose $P$; and $r_c^P$= timestamp of the position or velocity command at pose $P$. |
| **Unit of measure** | seconds |
| **Notes** | The timestamp of actuation initiation, $r_a$, is found by monitoring the “motor halt” flag after a new position or velocity value has been sent to the “target” registers.<br>The timestamp of the position command, $r_c$, is found by logging the time the first command packet for the actuation is transmitted from the robot controller to the robot driver. |

| | |
|---|---|
| **ID** | 4.2 |
| **Name** | Pose Travel Time ($R_P$) |
| **Description** | Amount of time for the robot arm to move between two poses |
| **Formula** | $R_L^{P_{i\to j}} = r_b^{P_j} - r_a^{P_i}$ ,<br>where $R_L^{P_{i\to j}}$= travel time required to move between pose $P_i$ and the next pose $P_j$; $r_b^{P_j}$= timestamp of actuation completion at pose $P_j$; and $r_a^{P_i}$= timestamp of actuation initiation at pose $P_i$. |
| **Unit of measure** | seconds |
| **Notes** | The timestamp of actuation completion ($r_b$) is found by monitoring the "motor halt" flag and "target" registers in each joint motor controller, and timestamping the transition of all joint flags from FALSE to TRUE after the register has been modified.<br>The timestamp of actuation initiation ($r_a$) is found by monitoring the "motor halt" flag and "target" registers in each joint motor controller, and timestamping the transition of any joint flag from TRUE to FALSE after the register has been modified. |

| | |
|---|---|
| **ID** | 4.3 |
| **Name** | Position Accuracy ($R_A$) |
| **Description** | Deviation between the commanded position and the measured position |
| **Formula** | $R_A = \sqrt{(\overline{x} - x_c)^2 + (\overline{y} - y_c)^2 + (\overline{z} - z_c)^2}$ ,<br>where<br>$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$<br>$\overline{y} = \frac{1}{n}\sum_{k=1}^{n} y_k$<br>$\overline{z} = \frac{1}{n}\sum_{k=1}^{n} z_k$ ,<br>where $x_c$, $y_c$, and $z_c$ = coordinates of the commanded pose; $x_k$, $y_k$, and $z_k$ = coordinates of the $k$-th attained pose; and $n$= number of measurement samples. |
| **Unit of measure** | millimeters |

| | |
|---|---|
| **ID** | 4.4 |

| | |
|---|---|
| **Name** | Position Repeatability ($R_R$) |
| **Description** | Closeness of agreement between the measured position after *n* repeat visits to the same commanded position |
| **Formula** | $R_R = \bar{l} + 3S_l$ ,<br>where<br>$\bar{l} = \frac{1}{n}\sum_{k=1}^{n} l_k$<br>$l_k = \sqrt{(x_k - \bar{x})^2 + (y_k - \bar{y})^2 + (z_k - \bar{z})^2}$<br>with $x$, $y$, $z$ and $x_k$, $y_k$, $z_k$ as defined in 4.3; and<br>$S_l = \sqrt{\frac{\sum_{k=1}^{n}(l_k - \bar{l})^2}{n-1}}$. |
| **Unit of measure** | millimeters |

| | |
|---|---|
| **ID** | 4.5 |
| **Name** | Energy Consumption ($R_E$) |
| **Description** | Estimation of the amount of power consumed by the robot joint motors |
| **Formula** | $R_E^T = \sum_{h=1}^{H} \int_{t_0}^{t_i} j_I^h(t)^2 \Omega_h \, dt$<br>with<br>$T = t_0 \leq t < t_i$ ,<br>where $R_E^T$= energy consumption during the timespan $T$; $H$= total number of joints; $t_0$= initial sample time; $t_i$= final sample time; $j_I^h(t)^2$= square of the current at joint $h$ at time $t$; $\Omega_h$= resistance of the motor terminal at joint $h$. |
| **Unit of measure** | Joules |
| **Notes** | Terminal resistance $\Omega_j$ of the joint motors 1 through 5 are defined for each joint as: 0.978 Ω, 0.978 Ω, 0.978 Ω, 4.48 Ω, 13.7 Ω, respectively. |

|  |  |
|---|---|
| **ID** | 5.1 |
| **Name** | Packet Headers ($n_H$) |
| **Description** | Header data of a captured packet |
| **Notes** | Headers may include appended data from the tools used to capture the packet. |

|  |  |
|---|---|
| **ID** | 5.2 |
| **Name** | Packet Data ($n_D$) |
| **Description** | Application layer payload of a captured packet |
| **Notes** | Allows monitoring of multiple subsystems within the enclave, and enables inspection of the enclave operations across multiple nodes. |

|  |  |
|---|---|
| **ID** | 5.3 |
| **Name** | Packet Protocol Type ($n_K$) |
| **Description** | Protocol type of a captured packet |
| **Notes** | Protocol types of captured packets may change based on the applied cybersecurity technologies for each individual experiment. |

|  |  |
|---|---|
| **ID** | 5.4 |
| **Name** | Packet Timestamp ($n_T$) |
| **Description** | Timestamp of a captured packet |
| **Unit of measure** | seconds |
| **Notes** | Post-processed timestamps have their epoch converted to the start time of the experiment. Raw data timestamps from an enclave experiment are measured relatively from the Unix epoch. |

| | | |
|---|---|---|
| **ID** | 5.5 | |
| **Name** | Packet Size ($n_S$) | |
| **Description** | Size of a packet in bytes | |
| **Unit of measure** | bytes | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 65535 |

| | |
|---|---|
| **ID** | 5.6 |
| **Name** | Packet Errors Counter ($n_E$) |
| **Description** | Number of packets transmitted or received with errors |

| | |
|---|---|
| **ID** | 5.7 |
| **Name** | Dropped Packets Counter ($n_D$) |
| **Description** | Number of packets dropped by the network |

| | |
|---|---|
| **ID** | 5.8 |
| **Name** | Packet Counter ($n_C$) |
| **Description** | Number of packets transmitted and received |
| **Notes** | Measurement may be in reference to enclave, between nodes, or to/from a single node. |

| | |
|---|---|
| **ID** | 6.1 |
| **Name** | Packet Path Delay ($N_D$) |
| **Description** | Time delay along the path from transmitter to receiver |
| **Formula** | $N_D^{p,a\to b} = n_T^{p,b} - n_T^{p,a}$ ,<br>where $N_D^{p,i\to j}$= path delay for packet $p$ from node $i$ to $j$; $n_T^{p,b}$= timestamp of packet $p$ when received at node $b$; $n_T^{p,j}$= timestamp of packet $p$ when transmitted at node $a$. |
| **Unit of measure** | seconds |
| **Notes** | Path delay calculations are especially useful when cybersecurity hardware is added to the wire, allowing accurate measurement of any network performance impacts. |

| | |
|---|---|
| **ID** | 6.2 |
| **Name** | Inter-packet Delay ($N_J$) |
| **Description** | Difference between the packet path delay of two packets |
| **Formula** | $N_J^{p_i} = N_D^{p_i} - N_D^{p_{i-1}}$ ,<br>where $N_J^{p_i}$= time delay between packet $p_i$ and the previous packet; $N_D^{p_i}$= packet path delay for packet $p_i$; and $N_D^{p_{i-1}}$= packet path delay for packet $p_{i-1}$. |
| **Unit of measure** | seconds |
| **Notes** | When using this KPI to described the inter-packet delay variation for a series of packets, the result is typically communicated by minimum and maximum values, mean and standard deviation, or a histogram. |

| | |
|---|---|
| **ID** | 6.3 |
| **Name** | TCP Packet Round Trip Time ($N_R$) |
| **Description** | Amount of time for the source node of a packet to receive the acknowledgement of receipt (ACK) from the destination node |
| **Formula** | $N_R = n_T^a - n_T^p$ ,<br>where $n_T^p$= timestamp of packet $p$ when transmitted; and $n_T^a$= timestamp of received ACK for packet $p$. |
| **Unit of measure** | seconds |
| **Notes** | This KPI is also known as round-trip time (RTT). |

| | | |
|---|---|---|
| **ID** | 6.4 | |
| **Name** | Information Ratio ($N_I$) | |
| **Description** | Ratio of the quantity of process information packets to all packets | |
| **Formula** | $N_I = \frac{n_I}{n_C}$ ,<br>where $n_I$= number of information packets. | |
| **Unit of measure** | % | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 100 |
| **Notes** | Information packets are defined as: any packet containing information that is used to operate the manufacturing process. | |

| | |
|---|---|
| **ID** | 6.5 |
| **Name** | Bit Rate ($N_B$) |
| **Description** | Rate of bits transmitted or received over a specific timespan |
| **Formula** | $\overline{N_B} = \frac{8}{t_i - t_0} \sum_{t \in T} n_S^t$<br>with<br>$T = \{x \in \mathbb{S} : t_0 < x \leq t_i\}$ ,<br>where $n_S^t$= the size of the packet at time $t$ in bytes; $\mathbb{S}$= set of captured packets; and $t_0, t_i$= initial and final time, respectively, of the desired timespan. |
| **Unit of measure** | bits / second |
| **Notes** | Specific derivatives of the KPI may be calculated as the average bits per second over the total experiment time, or calculated over discrete intervals. |

| | |
|---|---|
| **ID** | 6.6 |
| **Name** | Packet rate ($N_P$) |
| **Description** | Rate of packets transmitted and received over a specific amount of time |
| **Formula** | $$N_P = \frac{n_C^{t_i} - n_C^{t_0}}{t_i - t_0} \quad ,$$ where $n_C^{t_i}$= number of packets sent or received at time $t_i$; $n_C^{t_0}$= number of packets sent or received at time $t_0$; $t_i$= measurement period end time in seconds; and $t_0$= measurement period start time in seconds. |
| **Unit of measure** | packets / second |
| **Notes** | Specific derivatives of the KPI may be calculated as the average packets per second over the total experiment time, or calculated over discrete intervals. |

| | |
|---|---|
| **ID** | 6.7 |
| **Name** | Packet Error Rate ($N_E$) |
| **Description** | Rate of packets transmitted or received with errors over a specific amount of time |
| **Formula** | $$N_E = \frac{n_E^{t_i} - n_E^{t_0}}{t_i - t_0} \quad ,$$ where $n_E^{t_i}$= number of packets sent or received with errors at time $t_i$; $n_E^{t_0}$= number of packets sent or received with errors at time $t_0$; $t_i$= measurement period end time in seconds; and $t_0$= measurement period start time in seconds. |
| **Unit of measure** | packets / second |
| **Notes** | Specific derivatives of the KPI may be calculated as the average number of packets with errors per second over the total experiment time, or calculated over discrete intervals. |

| | | |
|---|---|---|
| **ID** | 6.8 | |
| **Name** | Proportion of Protocol Type ($N_T$) | |
| **Description** | Numerical proportion of a unique packet protocol type observed | |
| **Unit of measure** | % | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 100 |
| **Notes** | Captured packet protocol types included in this KPI may change based on the applied cybersecurity technologies for each individual experiment. | |

| | |
|---|---|
| **ID** | 7.1 |
| **Name** | Task Execution Time ($m_E$) |
| **Description** | Time required to complete the PLC task |
| **Unit of measure** | seconds |
| **Notes** | PLC task execution time data is recorded as the average task execution time of each cycle over a one second period. |

| | |
|---|---|
| **ID** | 7.2 |
| **Name** | Minimum Task Execution Time ($m_F$) |
| **Description** | Minimum amount of time required to complete the PLC task |
| **Unit of measure** | seconds |

| | |
|---|---|
| **ID** | 7.3 |
| **Name** | Maximum Task Execution Time ($m_G$) |
| **Description** | Maximum amount of time required to complete the PLC task |
| **Unit of measure** | seconds |

| | | |
|---|---|---|
| **ID** | 7.4 | |
| **Name** | CPU Utilization ($m_T$) | |
| **Description** | Amount of PLC CPU usage | |
| **Unit of measure** | % | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 100 |
| **Notes** | PLC CPU utilization data is recorded by the PLC every 100 milliseconds. | |

| | | |
|---|---|---|
| **ID** | 7.5 | |
| **Name** | Maximum CPU Utilization ($m_V$) | |
| **Description** | Maximum amount of PLC CPU usage | |
| **Unit of measure** | % | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 100 |
| **Notes** | PLC CPU utilization data is recorded by the PLC every 100 milliseconds. | |

| | |
|---|---|
| **ID** | 8.1 |
| **Name** | Task Execution Time Distribution ($M_E$) |
| **Description** | Mean and standard deviation of PLC task execution time |
| **Formula** | $M_E = \overline{m_E} \pm \sigma_{m_E}$ |
| **Unit of measure** | Seconds |

| | |
|---|---|
| **ID** | 8.2 |
| **Name** | CPU Utilization Distribution ($M_T$) |
| **Description** | Mean and standard deviation of PLC CPU utilization |
| **Formula** | $M_E = \overline{m_T} \pm \sigma_{m_T}$ |
| **Unit of measure** | % |
| **ID** | 9.1 |

| | |
|---|---|
| **Name** | CPU Utilization Timers ($c_U$) |
| **Description** | Amount of server CPU utilization |
| **Unit of measure** | seconds |
| **Notes** | Metric is recorded as the total amount of utilization seconds across all CPUs within the server [8], [9]. The CPU utilization modes recorded for the metric are:<br>• **user** – amount of time the CPU was executing non-kernel processes<br>• **system** – amount of time the CPU was executing kernel processes<br>• **idle** – amount of time the CPU had nothing to process<br>• **iowait** – amount of time the CPU was waiting for an I/O operation to complete (e.g. hard drive read or write)<br>• **irq** – amount of time the CPU was servicing a hardware interrupt<br>• **softirq** – amount of time the CPU was servicing a software interrupt<br>• **steal** – amount of time a virtualized CPU was waiting to use the CPU<br>• **guest** – amount of time the CPU spent executing processes for a hosted virtual machine<br>• **guest**_nice – amount of time the CPU spent executing a niced process for a hosted virtual machine (see *nice*)<br>• **nice –** amount of time the CPU spent executing a process with a modified execution priority (can be either a higher or lower priority)<br>The metric mode used in KPI equations will be denoted by a second subscript. For example, the system CPU time is denoted as $c_{U_{system}}$. |

| | |
|---|---|
| **ID** | 9.2 |
| **Name** | Available Memory ($c_A$) |
| **Description** | Amount of server memory available by the server |
| **Unit of measure** | bytes |

| | |
|---|---|
| **ID** | 9.3 |
| **Name** | Total Memory ($c_M$) |
| **Description** | Total amount of memory installed in the server |
| **Unit of measure** | bytes |
| **Notes** | This metric is measured once at the beginning of the experiment. |

| | |
|---|---|
| **ID** | 9.4 |

| | |
|---|---|
| **Name** | Disk I/O Byte Counters ($c_F$) |
| **Description** | Number of bytes read and written to the server hard drive(s) |
| **Unit of measure** | bytes |
| **Notes** | Each measurement has a read and write component:<br>$c_F = (c_{FR}, c_{FW})$ ,<br>where the subscript (R or W) defines whether the metric describes disk reads or writes. |

| | |
|---|---|
| **ID** | 9.5 |
| **Name** | Disk I/O Counters ($c_G$) |
| **Description** | Number of discrete read and write operations to the server hard drive(s) since the last sample |
| **Notes** | Each measurement has a read and write component:<br>$c_G = (c_{GR}, c_{GW})$ ,<br>where the subscript (R or W) defines whether the metric describes disk reads or writes. |

| | |
|---|---|
| **ID** | 9.6 |
| **Name** | Disk I/O Access Time ($c_H$) |
| **Description** | Amount of time used to read and write to the server hard drive(s) since the last sample |
| **Unit of measure** | seconds |
| **Notes** | Each measurement has a read and write component:<br>$c_H = (c_{HR}, c_{HW})$ ,<br>where the subscript (R or W) defines disk reads or writes, respectively. |

| | |
|---|---|
| **ID** | 9.7 |
| **Name** | Network Byte Counters ($c_B$) |
| **Description** | Number of bytes transmitted and received by the server |
| **Unit of measure** | bytes |
| **Notes** | Each measurement has a transmit and receive component:<br>$$c_B = (c_{BT}, c_{BR}) \quad ,$$<br>where the subscript (T or R) defines network transmission or receipt, respectively. |

| | |
|---|---|
| **ID** | 9.8 |
| **Name** | Network Packet Counters ($c_P$) |
| **Description** | Number of packets transmitted and received by the server |
| **Unit of measure** | bytes |
| **Notes** | Each measurement has a transmit and receive component:<br>$$c_P = (c_{PT}, c_{PR}) \quad ,$$<br>where the subscript (T or R) defines network transmission or receipt, respectively. |

| | |
|---|---|
| **ID** | 9.9 |
| **Name** | Network Packet Errors ($c_E$) |
| **Description** | Number of packets with errors transmitted and received by the server |
| **Notes** | Each measurement has a transmit and receive component:<br>$$c_E = (c_{ET}, c_{ER}) \quad ,$$<br>where the subscript (T or R) defines network transmission or receipt, respectively. |

| | |
|---|---|
| **ID** | 9.10 |
| **Name** | Network Dropped Packets ($c_X$) |
| **Description** | Number of incoming or outgoing packets dropped by the server |
| **Notes** | Each measurement has a transmit and receive component:<br>$$c_X = (c_{XT}, c_{XR}) \quad ,$$<br>where the subscript (T or R) defines network transmission or receipt, respectively. |

<table>
<tr><td><b>ID</b></td><td colspan="2">10.1</td></tr>
<tr><td><b>Name</b></td><td colspan="2">CPU Utilization ($C_U$)</td></tr>
<tr><td><b>Description</b></td><td colspan="2">Mean and standard deviation of server CPU utilization</td></tr>
<tr><td><b>Formula</b></td><td colspan="2">
$$C_U^T = \overline{c_U^T} \pm \sigma_U^T \quad ,$$
where
$$\overline{c_U^T} = \frac{1}{|T|} \sum_{t \in T} c_{U_x}^t$$
$$\sigma_U^T = \sqrt{\frac{1}{|T|-1} \sum_{t \in T} \left(c_{U_x}^t - \overline{c_U^T}\right)^2} \quad ,$$
where
$$c_{U_x} = c_{U_{user}} + c_{U_{system}}$$
$$T = \{x \in \mathbb{S} : (t_0 + \delta) < x \leq t_i\} \quad ,$$
where $c_{U_{user}}$= measured amount of user CPU utilization; $c_{U_{system},t}$= measured amount of system CPU utilization; $|T|$= cardinality of the set $T$; $\mathbb{S}$= set of server performance measurement timestamps; $\delta$= sample period of the logger in seconds; and $t_0, t_i$= initial and final time, respectively, of the desired timespan.
</td></tr>
<tr><td><b>Unit of measure</b></td><td colspan="2">%</td></tr>
<tr><td rowspan="2"><b>Range</b></td><td><b>Minimum</b></td><td>0</td></tr>
<tr><td><b>Maximum</b></td><td>100</td></tr>
<tr><td><b>Notes</b></td><td colspan="2">Specific derivatives of the KPI may be calculated over the total experiment time, or calculated between specific time intervals.<br>The set $T$ does not include the nearest sample $t_0$ because the resulting calculation would include data that are outside of the desired timespan. To account for this, the sample period of the logger ($\delta$) is used to coerce the set to only include samples that have measured data within the specified timespan.</td></tr>
</table>

| | | |
|---|---|---|
| **ID** | 10.2 | |
| **Name** | Memory Utilization ($C_X$) | |
| **Description** | Mean and standard deviation of server memory utilization | |
| **Formula** | $C_X^T = \overline{c_X^T} \pm \sigma_X^T \quad ,$<br>where<br>$\overline{c_X^T} = \frac{1}{\lvert T\rvert}\sum_{t\in T} c_X^t$<br>$\sigma_X^T = \sqrt{\frac{1}{\lvert T\rvert - 1}\sum_{t\in T}\left(c_X^t - \overline{c_X^T}\right)^2} \quad ,$<br>where<br>$c_X^t = \left(\frac{c_M - c_A^t}{c_M}\right)$<br>$T = \{x \in \mathbb{S} : t_0 \le x \le t_i\} \quad ,$<br>where $c_{X,t}$= amount of utilized server memory at time $t$; $c_M$= total amount of server memory; $c_{A,t}$= total amount of available memory at time $t$; $\lvert T\rvert$= cardinality of the set $T$; $\mathbb{S}$= set of server performance measurement timestamps; and $t_0, t_i$= initial and final time, respectively, of the desired timespan. | |
| **Unit of measure** | % | |
| **Range** | **Minimum** | 0 |
| | **Maximum** | 100 |
| **Notes** | Specific derivatives of the KPI may be calculated over the total experiment time, or calculated between specific time intervals. | |

| **ID** | 10.3 |
|---|---|
| **Name** | Disk I/O Throughput ($C_F$) |
| **Description** | Mean rate and standard deviation of data read and written to the server hard drive(s) |
| **Formula** | $$C_F^T = \overline{c_F^T} \pm \sigma_F^T \quad ,$$<br>where<br>$$\overline{c_F^T} = \frac{1}{\lvert T \rvert} \sum_{t \in T} \frac{\Delta c_F^t}{\Delta t}$$<br>$$\sigma_F^T = \sqrt{\frac{1}{\lvert T \rvert - 1} \sum_{t \in T} \left( \frac{\Delta c_F^t}{\Delta t} - \overline{c_F^T} \right)^2} \quad ,$$<br>where<br>$$\Delta c_F^t = c_F^t - c_F^{t-1}$$<br>$$T = \{x \in \mathbb{S} : (t_0 + \delta) < x \leq t_i\} \quad ,$$<br>where $\Delta c_{F,t}$= change in disk I/O at time $t$ from the previous sample; $\Delta t$= change in time $t$ from the previous sample; $\mathbb{S}$= set of server performance measurement timestamps; $\delta$= sample period of the logger in seconds; and $t_0, t_i$= initial and final time, respectively, of the desired timespan. |
| **Unit of measure** | bytes / second |
| **Notes** | Specific derivatives of the KPI may be calculated over the total experiment time, or calculated between specific time intervals.<br>The set $T$ does not include the nearest sample $t_0$ because the resulting calculation would include data that are outside of the desired timespan. To account for this, the sample period of the logger ($\delta$) is used to coerce the set to only include samples that have measured data within the specified timespan.<br>Per definition 9.4 (Disk I/O Byte Counters), each measurement includes a read and write component:<br>$$c_F = \{(c_{FR}, c_{FW})\} \therefore C_F = (C_{FR}, C_{FW}).$$ |

| | |
|---|---|
| **ID** | 10.4 |
| **Name** | Network Bitrate ($C_B$) |
| **Description** | Mean rate and standard deviation of bits transmitted and received by the server |
| **Formula** | $C_B^T = \overline{c_B^T} \pm \sigma_B^T \quad ,$<br>where<br>$\overline{c_B^T} = \frac{8}{\lvert T\rvert}\sum_{t\in T}\frac{\Delta c_B^t}{\Delta t}$<br>$\sigma_B^T = \sqrt{\frac{1}{\lvert T\rvert - 1}\sum_{t\in T}\left(c_B^t - \overline{c_B^T}\right)^2} \quad ,$<br>where<br>$\Delta c_B^t = c_B^t - c_B^{t-1}$<br>$T = \{x \in \mathbb{S} : (t_0 + \delta) < x \le t_i\} \quad ,$<br>where $\Delta c_{B,t}$= change in network bytes at time $t$ from the previous sample; $\Delta t$= change in time $t$ from the previous sample; $\mathbb{S}$= set of server performance measurement timestamps; $\delta$= sample period of the logger in seconds; and $t_0, t_i$= initial and final time, respectively, of the desired timespan. |
| **Unit of measure** | bits / second |
| **Notes** | Specific derivatives of the KPI may be calculated over the total experiment time, or calculated between specific time intervals.<br>The set $T$ does not include the nearest sample $t_0$ because the resulting calculation would include data that are outside of the desired timespan. To account for this, the sample period of the logger ($\delta$) is used to coerce the set to only include samples that have measured data within the specified timespan.<br>Per definition 9.7 (Network Byte Counters), each measurement includes a transmit and receive component:<br>$c_B = \{(c_{BT}, c_{BR})\} \therefore C_B = (C_{BT}, C_{BR}).$ |

| | |
|---|---|
| **ID** | 10.5 |
| **Name** | Network Packet Rate ($C_P$) |
| **Description** | Mean rate and standard deviation of packets transmitted and received by the server |
| **Formula** | $C_P^T = \overline{c_P^T} \pm \sigma_P^T$ ,<br>where<br>$\overline{c_P^T} = \frac{1}{\lvert T\rvert}\sum_{t\in T}\frac{\Delta c_P^t}{\Delta t}$<br>$\sigma_P^T = \sqrt{\frac{1}{\lvert T\rvert - 1}\sum_{t\in T}\left(c_P^t - \overline{c_P^T}\right)^2}$ ,<br>where<br>$\Delta c_P^t = c_P^t - c_P^{t-1}$<br>$T = \{x \in \mathbb{S} : (t_0 + \delta) < x \le t_i\}$ ,<br>where $\Delta c_{P,t}$= change in network packet quantity at time $t$ from the previous sample; $\Delta t$= change in time $t$ from the previous sample; $\mathbb{S}$= set of server performance measurement timestamps; $\delta$= sample period of the logger in seconds; and $t_0$, $t_i$= initial and final time, respectively, of the desired timespan. |
| **Unit of measure** | packets / second |
| **Notes** | Specific derivatives of the KPI may be calculated over the total experiment time, or calculated between specific time intervals.<br>The set $T$ does not include the nearest sample $t_0$ because the resulting calculation would include data that are outside of the desired timespan. To account for this, the sample period of the logger ($\delta$) is used to coerce the set to only include samples that have measured data within the specified timespan.<br>Per definition 9.8 (Network Packet Counters), each measurement includes a transmit and receive component:<br>$c_P = \{(c_{PT}, c_{PR})\} \therefore C_P = (C_{PT}, C_{PR})$. |

# 6 APPENDIX B – REFERENCES

[1] R. Candell, T. Zimmerman, and K. Stouffer, “An Industrial Control System Cybersecurity Performance Testbed,” NISTIR 8089, National Institute of Standards and Technology (NIST), 2015.

[2] I. C. Garretson, M. Mani, S. Leong, K. W. Lyons, and K. R. Haapala, “Terminology to support manufacturing process characterization and assessment for sustainable production,” *J. Clean. Prod.*, vol. 139, pp. 986–1000, 2016.

[3] “Manipulating industrial robots - Performance criteria and related test methods.” ISO Standard 9283, 1998.

[4] “Automation systems and integration - Key performance indicators (KPIs) for manufacturing operations management.” ISO Standard 22400, 2014.

[5] R. Baroudi, *KPI Mega Library: 17,000 Key Performance Indicators*. Scotts Valley, California: CreateSpace Independent Publishing Platform, 2010.

[6] C. Demichelis and P. Chimento, “IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), RFC 3393,” *Internet Eng. Task Force*, 2002.

[7] H. Allen and C. Vincente, “Network Performance Definitions and Analysis,” in *PacNOG 5*, 2009.

[8] G. Rodola’, “psutil documentation -- psutil 5.1.2 documentation,” *pythonhosted.org*, 2017. [Online]. Available: http://pythonhosted.org/psutil/. [Accessed: 09-Feb-2017].

[9] D. Haynes, “Understanding Linux CPU Stats,” *ScoutApp.com*, 2015. [Online]. Available: http://blog.scoutapp.com/articles/2015/02/24/understanding-linuxs-cpu-stats. [Accessed: 09-Feb-2017].