Special Issue Article

JDMS

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
2022, Vol. 19(1) 23–36

DOI: 10.1177/1548512920960537
journals.sagepub.com/home/dms
SAGE

Cybersecurity threats and experimental testbed for a generator system

Aaron W Werth, SueAnne N Griffith, Jesse R Hairston and Thomas H Morris

Abstract

In this work, a high-fidelity virtual testbed modeling a networked diesel generator, similar to those used commercially and by the military, is described. This testbed consists of a physical system model of a generator, a digital control system, a remote monitoring system, and physical and networked connections. The virtual testbed allows researchers to emulate a cyber-physical system and perform cyber attacks against the system without the monetary and safety risks associated with a testbed created from physical components. The testbed was used to feasibly simulate network, hardware Trojan, and software Trojan attacks against the diesel generator, and to observe the cyber and physical outcomes.

Keywords

cybersecurity, generator system, attacks, vulnerabilities, modeling and simulation, supply chain

1. Introduction

Hospitals, businesses, and other commercial buildings often use networked generators to provide power to critical systems in the event of an electrical outage. Manufacturers lose money in the time they cannot produce, hospitals must have a continuous source of power to operate critical equipment, and nuclear power plants require a constant source of electricity to operate reactor coolant pumps. Because of their importance in times of need, networked generator systems are targets of interest to cyber criminals and nation states. A cyber attack could cause damage to the generator or its physically connected subsystems, leading to power loss and inability to use connected devices. Researchers have worked to understand the vulnerabilities and impacts of cyber attacks against systems using testbeds. Due to monetary and safety concerns, virtual testbeds are preferred over their physical counterparts for cyber-physical attack simulations. This approach has been shown effective in modeling cyber attacks on electrical systems.[1]

The focus of this work is the development of a virtual, modular testbed to provide a high- fidelity model of the cyber and physical components of a networked generator system that incorporates features lacking in similar testbeds. A modular virtualization approach was first introduced by Alves et al. in 2016 and further refined in 2018.[2,3] Use of a testbed allows development of high-fidelity testbeds without requiring a large laboratory setup. Testbeds consist of virtualized components, including a physical system model, a cyber-physical link, a controller, a virtual network, and a human machine interface. The goal of this present work is to build upon the work of Alves et al. to create a generator testbed for studying threat vectors, attack outcomes, and security controls without risk of damage to a costly physical system.[3]

Previous works include a publication by Korkmaz et al. that describes a testbed comprising a physical generator, programmable logic controller (PLC), and computer with a human machine interface.[4] Because the Korkmaz model is not virtual, however, it is expensive to reproduce, requires

Center for Cybersecurity Research and Education, The University of Alabama in Huntsville Huntsville, AL, USA

Corresponding author:
Thomas H Morris, Center for Cybersecurity Research and Education, The University of Alabama in Huntsville, 301 Sparkman Dr., Huntsville, AL 35899, USA.
Email: tommy.morris@uah.edu

continued maintenance, and cannot be easily shared. Ashok et al. modeled a microgrid, including multiple generators, loads, attack nodes, and a control center, using a Real Time Digital Simulator (RTDS) and Automatic Generation Control (AGC).[5] This testbed was used to study the impacts of network-borne attacks against the microgrid. The RTDS simulator provided the ability to model the physical system with high fidelity, but results in a costly testbed without portability.

In addition to the previous works on generator testbeds, which lack fully implemented virtualization, it is helpful to review works related to testbeds for power systems and other relevant systems. Researchers have developed multiple testbeds to simulate cyber attacks against power systems. The Testbed for Analyzing Security of SCADA Control Systems (TASSCS) is a sophisticated testbed developed to simulate the effects of network-based cyber attacks. TASSCS provides high fidelity simulation of SCADA networks using MODBUS and DNP3 protocols. The testbed's network is simulated through the network library Opnet. TASSCS does not simulate a complete PLC or a digital control system (DCS). Therefore, DCS vulnerabilities cannot be tested.[6] In contrast, the approach in this present work does incorporate a DCS for greater fidelity. Adhikari et al. developed a testbed for bulk electric transmission systems with wide area measurement components using a RTDS and hardware-in-the-loop protection relays, phasor measurement units, and phasor data concentrators.[7] The testbed demonstrated its use for the study of faults and network-based cyber attacks, but did not focus on a specific generator subsystem with high fidelity.

For virtual testbeds in general, Reaves and Morris developed a testbed framework for data collection and intrusion detection research.[8] Their testbed was built to study a high-fidelity gas pipeline, but the framework may be used for other types of cyber-physical systems, such as power systems. Although the testbed developed by Reaves and Morris is modular and portable, Python scripts were used as controllers instead of virtualizing an actual DCS, and physical system components were modeled with simple equations derived from curve fits of behaviors observed from a laboratory scale gas pipeline.[8] The Reaves testbed does not match true physical system behavior when the simulated system state deviates from state initially observed on the laboratory scale reference system.

Lastly, Cintuglu et al. published a survey of power system testbeds to model ARP spoofing, denial of service (DoS), malware, and malformed packet attacks.[9] Many testbeds surveyed are virtual; however, they focus on the power grid as a whole, not a generator subsystem and its major components necessary for studying exploits. Some of the testbeds incorporate simple generators, but only as a basic power source. Other surveyed testbeds use non-virtual generators.

This current work offers four major contributions. First, a generator testbed was created that is fully virtualized. A model of the physical system was built in MATLAB Simulink and based upon Yeager and Willis' work in modeling emergency diesel generators.[10] Sensors were added to the model, and the generator's governor, a controller that regulates the speed of the engine, was removed from the MATLAB Simulink model and implemented in ladder logic running on the generator's digital control system as a discrete control algorithm. Second, a higher-fidelity model of the generator was included, which allows for more types of threat models to be evaluated. The higher fidelity of representing the governor as a digital control algorithm implemented as software in the DCS enables the study of cyber attacks that affect parameters of the algorithm. The digital control system has high fidelity in representing a real generator system's functions with regard to monitoring of sensor data and response to faults. A fault is defined as an abnormal occurrence or problem due to damaged equipment.[11] This allows for other cyber attack exploits that would trigger faults to be studied. Third, this work examines case studies of cyberattacks including two supply chain attacks: a simulated hardware Trojan that is triggered by a time bomb, and a software Trojan activated by a logic bomb. Network-based cyber attacks were also performed. These cyber attacks were modeled in order to observe the effect that they would have on a generator system. Fourth, a novel attack against the gain parameter of the generator's control system, causing chaotic physical behavior, was included and is unique from the well-known Aurora attack against a generator system by affecting the DCS software directly.[12] Table 1 summarizes the major contributions.

The remainder of this work is organized as follows: Section 2 covers the testbed's various modules and their implementations; Section 3 describes a selected set of attacks that were performed on the testbed; and Section 4 is the conclusion and discusses future work.

2. Testbed

This generator testbed was created using a modular approach involving the five primary components of a Supervisory Control and Data Acquisition (SCADA) system according to a framework presented in recent work.[2] These main components are (1) the physical system, (2) a cyber-physical link, (3) the Distributed Control System or DCS, (4) a network, and (5) the remote monitoring or control also known as a Human Machine Interface (HMI). Figure 1 illustrates this architecture.

Each of these modules will be described in depth in the following subsections along with their specific implementations for the generator system testbed.

Table 1. Major contributions of this work.

No.	Contribution	Description
1	Overall virtual generator Testbed	A generator testbed created that is fully virtualized as opposed to an actual physical implementation.
2	Higher-fidelity control algorithm	Higher fidelity of representing the governor as a digital control algorithm implemented as software in the DCS.
3	Two supply-chain attacks	(1) Simulated hardware Trojan triggered by a time bomb, (2) Software Trojan activated by a logic bomb
4	Novel cyber-physical attack	Command attack against the gain parameter of the generator's control system, inducing chaotic physical behavior

DCS: digital control system.

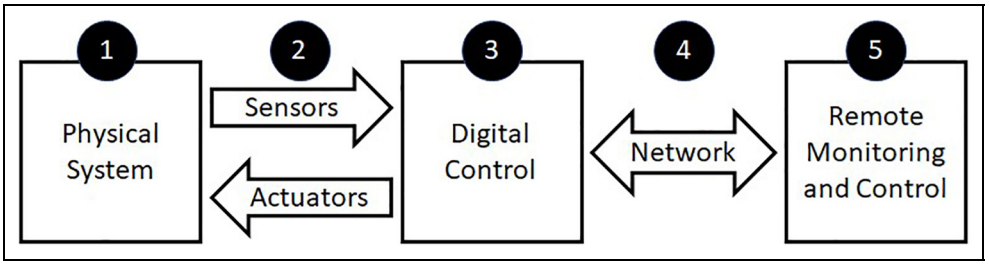

Figure 1. Modular structure of a cyber-physical system.

2.1. Physical system

The physical system in this testbed consists of a MATLAB Simulink model that simulates the basic operation of the electromechanical aspects of the generator. This model is based upon an example generator model designed by Yeager and Willis and provided in the MATLAB software.[10] The model in Yeager and Willis was rated at 2400 volts line-to-line and 3.25 MVA (megavolt amps).[10] These characteristics remained unchanged for the purposes of this testbed. The example was modified to include sensors to measure RPM (revolutions per minute), oil pressure, fuel level, and temperature for the engine as well as the output power frequency, V_{RMS} (voltage root mean square), and I_{RMS} (current root mean square). The generator in this work is connected to a simple load, such as a motor for running pumps. A model of a fuel tank was also added as a subsystem to represent the fuel as if it were in a storage tank similar to the case study described in Alves et al.[2] Only, in this case, Q_{in} is set to 0, and the Q_{out} is assumed to be 1.04 gal/h, which is the average fuel rate of consumption for the generator when it is on. It is also assumed that there is an initial amount of fuel when the generator is first switched on. To represent the thermodynamics of an engine, which varies with the RPM, a MATLAB model was integrated with the main generator model.[13] The resulting RPM produced from the generator's electromechanical model is fed as input into the thermodynamic model. By incorporating the thermodynamic model, the dynamic behavior of the temperature of the engine can be better represented. Additionally, the original example generator model contained a control system for the engine's governor, which regulates the fuel injection within the engine. The rate at which fuel is injected impacts the speed at which the engine operates, and, therefore, the frequency because the generator is a synchronous machine. This control system was removed from the Simulink model and redesigned in the DCS's ladder logic in order to better emulate a real system and allow for a logical separation of the physical model and its digital control system, which is normally implemented on a computer or embedded system in industry. Also, the physical system incorporated a set of actuators and sensors that communicate with the digital control system; these are listed in Table 2 in addition to related internal state registers.

2.2. Cyber-physical link

The cyber-physical link facilitates communication between the physical system and the DCS to allow for the transmission of sensor data from the physical system to the DCS, and also actuator commands from the DCS to the physical system. For the purposes of the testbed, this link is implemented as User Datagram Protocol (UDP) packets sent

Table 2. Generator Instrumentation – registers associated with sensors and actuators.

	Registers	Description	Range
Sensors	Vrms	Output voltage RMS	-
	Irms	Output current RMS	-
	Frequency	Output frequency	59.5 - 60.5 Hz. (49.5 - 50.5 Hz.)
	RPM	Revolutions per minute of engine	0 – 20,000 RPM
	Fuel level	Engine fuel level	0 - 100%
	Oil pressure	Engine oil pressure	10 PSI - 90 PSI
	Temperature	Engine temperature	0 - 300°C
Actuators	Com_Act	Fuel Injection	-
	Onoff	Toggle generator	{0, 1}
	Breaker	Connection to load though breaker	{0, 1}
Internal	Override	If enabled, prevents shutdown from faults	{0, 1}
	Fault	Engine state	{0, 1}
	Gain	Gain used in engine governor control system	-
	Freq. Switch	Toggle between 60 and 50 Hz output	{0, 1}

over a virtual network that connects the host machine on which the MATLAB simulation is running to the virtual machine that serves as the DCS. The purpose of this connection is not to emulate a network, but to use the send and forget property of UDP to emulate a wired connection transferring information between the DCS and physical system. Each of the sensors and actuators from Table 2 is assigned a UDP port on the DCS, and data to and from the physical model is sent and received by the corresponding port. This is implemented on the physical system using the Linux UDP Send and Receive blocks built into MATLAB Simulink. In this virtualization approach, the UDP packets sent by the physical model are relayed by an interface program running on the DCS virtual machine to the main DCS software. The values passed are then stored in registers in the DCS. This is the approach described in depth by Alves et al.[1,2]

2.3. Digital control system

The DCS is implemented on a Linux virtual machine equipped with OpenPLC, an open-source PLC (programmable logic controller) software package that executes control code written in ladder logic, an IEC-61131-3 standard language.[14] This DCS serves to monitor the sensor data, including the temperature of the engine and the speed of the engine shaft in RPM, as well as other information. Additionally, the DCS regulates the speed of the engine by means of a control algorithm, and control signals are sent from the DCS to actuators on the physical system to dictate engine speed and other operating conditions.

Figure 2 depicts the portion of the main ladder logic of the DCS, which controls the engine governor and checks the validity of sensor data. The sensor readings are passed through comparator blocks within the ladder logic; if these values leave a valid range, the DCS triggers a fault because there is a potential for the system to be physically damaged. Because safe operating ranges vary from system to system, these valid ranges are stored within the DCS as editable registers to allow operators to set the variables as needed. What follows is an explanation of each of the rungs in the ladder logic beginning at the top and proceeding to the bottom.

The purpose of the first and second rungs are for detecting if the voltage or current exceeds a maximum voltage and current, respectively. In such events, the generator triggers a fault. The third rung of the ladder logic shown uses both a less than block (LT) to compare the oil pressure (Oil) with the minimum (oil_min), and a greater than block (GT) to compare the pressure with the maximum pressure (oil_max). If either block's output is ''true'', the physical model's oil pressure is outside the safe operating range, and a fault is triggered by setting (S), the Boolean fault variable, to a value of ''true.'' The fourth rung down uses a greater than block to verify that the current engine temperature (Temp) has not risen above the maximum temperature (temp_max). If the temperature is too high, a fault is triggered. In the fifth rung of logic, the RPM is compared with the fastest allowable engine speed (RPM_max) via a greater than block. The fifth rung checks the Fault and Override variables to determine the generator's state (onoff). If no fault has been triggered (Fault = ''false'') and the fault override has not been set (Override = ''false''), the generator will remain on. If a fault is triggered and the override is not set, however, the generator will cut off. On other hand, the generator will remain on if the override is set despite any faults that occur. Also, the last rung's purpose is for the fault to be reset when desired by the user if a fault has occurred. Below these rungs is a selector block, which allows one of two given values for the reference to pass as the value for

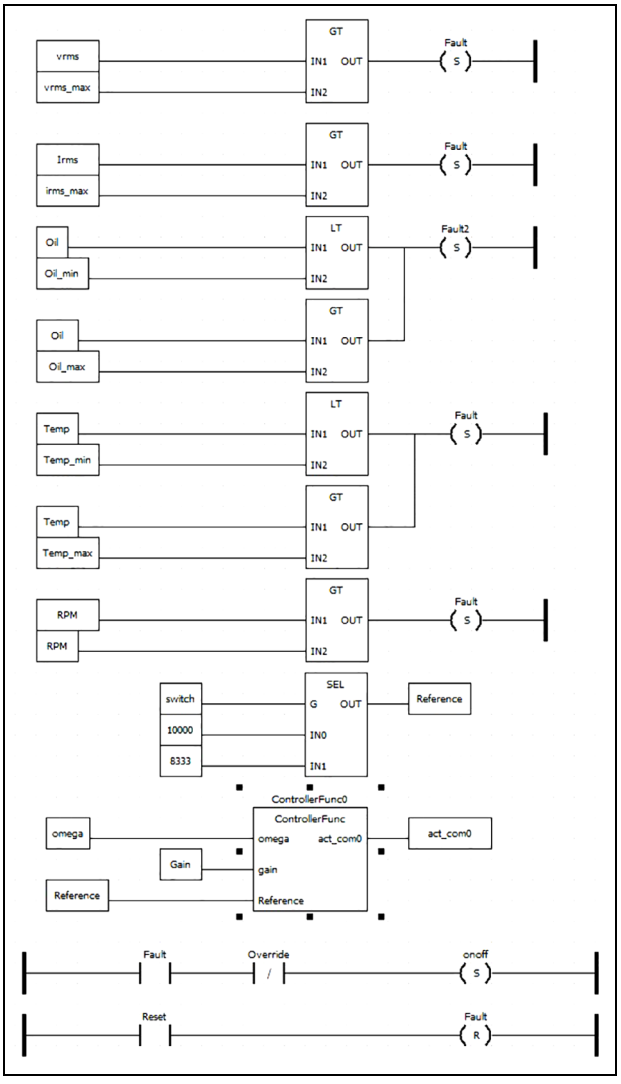

Figure 2. Ladder Logic, including the Governor Controller.
GT: greater than; irms: current root mean square; irms_max: maximum current root mean square; Oil: oil pressure; oil_max: maximum oil pressure; oil_min: minimal oil pressure; RPM: revolutions per minute; vrms: voltage root mean square; vrms_max: maximum voltage root mean square.

reference depending on the user's selection (boolean variable ''switch'') of the set point for frequency (60 Hz or 50 Hz).

In addition, included in the ladder logic is a block that serves as the controller, written in ST (Structured Text) code. This code is a high-level programming language as part of the IEC-61131-3 standard. An algorithm was created in this controller that implements the digital controller for regulating the speed of the engine. As seen at the bottom of Figure 2, the controller is implemented as a block in the ladder logic. This block takes the angular velocity (omega in the ladder logic) of the engine and the reference angular velocity as input. The controller then determines, based upon the transfer function, whether to increase, decrease, or maintain the speed at which fuel is injected into the engine. This value is output as the variable act_com in Figure 2.[10] This bottom subsystem (as seen in Figure 2) of the DCS uses a function block to implement a governor to regulate the speed of the engine shaft by controlling the fuel injection rate. This block is functionally equivalent to the dashed box in Figure 3.

W_{ref} (a variable that corresponds with ''Reference'' in the ladder logic) represents the reference, or desired, angular velocity of the engine shaft, while W (a variable that corresponds with Omega in the ladder logic) represents the current angular velocity. W is read by a sensor and passed through a zero-order hold, which is needed to convert the resulting value from a continuous-time value to a discrete one. Within the portion of the governor implemented in the DCS, the current velocity, which is given from sampled sensor readings, is subtracted from the reference velocity to determine the error, or the difference between the ideal and actual values. The resulting value is then passed through a discrete transfer function as represented as a block within the dashed area of Figure 3. This transfer function uses Equation 1:

$$H_d = \frac{9.518z - 8.448}{z^2 - 0.2977z + 0.3679} \quad (1)$$

The output of that function is multiplied by the gain, an adjustable parameter stored in a register within the DCS. The output of the gain is transmitted to the physical model and passed through the actuator block to produce the torque value, which is multiplied by W, the angular velocity. The product of this multiplication is P_m, the mechanical power, which is used by the Simulink model to determine the electrical output of the engine. The work by Yeager and Willis describes a continuous-time transfer function for this governor,[10] which this work discretized so that the transfer function can be implemented on the DCS. Because the DCS is a digital system, it is important to note that the DCS has a sampling period. Essentially, an input to the system is sampled at a period determined appropriate by the designer of the system. Updates to internal state variables and outputs, which may be determined partially by the inputs, are performed at the sampling period. If a higher sampling rate is selected, then the digital transfer function will behave more accurately like its continuous-time counterpart. Parameters of the transfer function were determined for a digital system with a sampling period of 20 ms. Special methods can be applied to convert a continuous transfer function to a discrete transfer function at a given sampling period. MATLAB provides commands that can carry out this operation.[15]

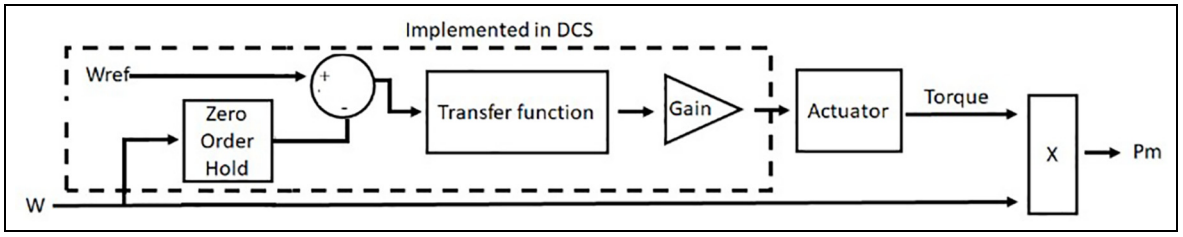

Figure 3. Block diagram of implementation of transfer function in Equation 1.
DCS: digital control system; Pm: mechanical power; W: variable corresponding with Omega in the ladder logic; Wref: variable corresponding with "Reference" in the ladder logic.

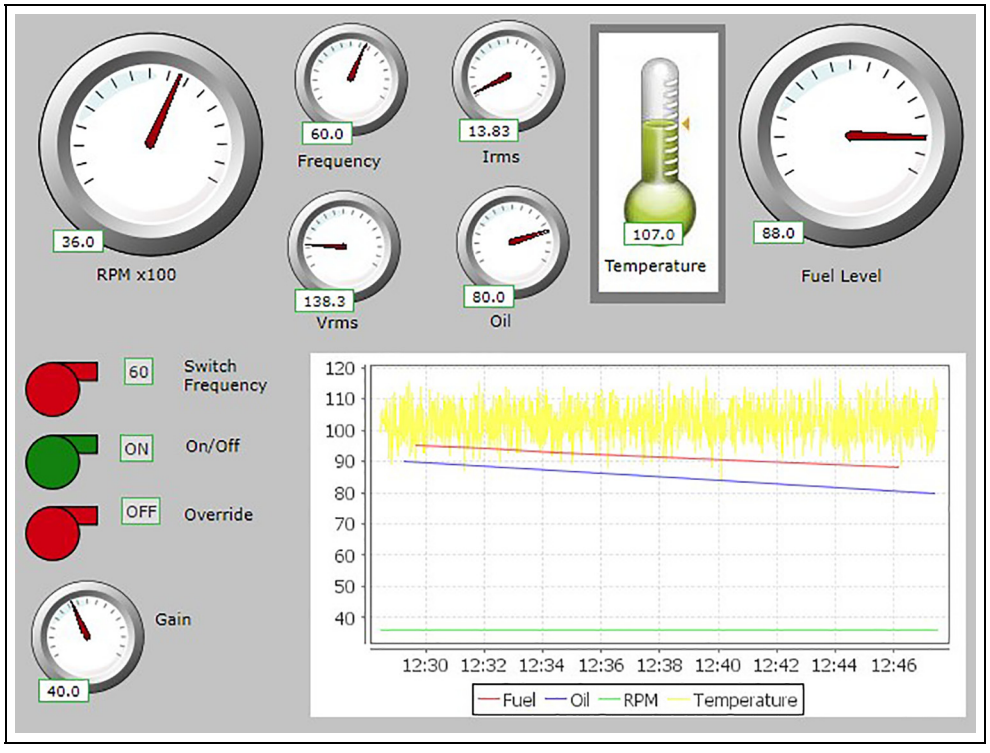

Figure 4. HMI designed for the generator model.
HMI: human machine interface; irms: current root mean square; RPM: revolutions per minute; vrms: voltage root mean square.

2.4. Network

The network between the testbed's DCS and the HMI is implemented as a virtual network. Communication between these systems is done using the MODBUS TCP/IP protocol, which is commonly used in ICSs. To have other computers, other virtual machines may be added to the network. These additional computers could include a threat actor or control devices for other networked systems.

2.5. Human machine interface

The HMI, shown in Figure 4, is used to allow a user to monitor and manage the DCS over a network. For this testbed, it is built using ScadaBR, an open source program designed for HMI production, and run in an internet browser window on the host computer.

In the model's HMI, generator sensor data is displayed through simulated analog dials and digital displays; both types of displays include the exact sensor reading in their bottom left. The chart in the bottom right of the HMI displays sensor readings over time, allowing the operator to observe history and trends in fuel and oil levels, RPM, and temperature.

Across the top of the HMI, there are two large dials: one for the generator's RPM and one for its fuel level as a percentage. The four smaller dials are, clockwise from top

Table 3. Modbus function codes and the value types requested.

Code to Read	Code to Write	Value Read	Data Type
01	05	Coil status	Digital output (1 bit)
02	N/A	Input status	Digital input (1 bit)
03	06	Holding register	Analog output (16 bit)
04	N/A	Input register	Analog input (16 bit)

left, the output frequency, current root mean square (I_{RMS}), voltage root mean square zero (V_{RMS0}, an adjusted V_{RMS}), and voltage root mean square (V_{RMS}). To the right of these small dials is the generator's temperature gauge, displayed as a thermometer with nine different temperature levels. In the HMI's bottom left, there are three sets of buttons and displays; these are for switching the generator's frequency, on/off status, and override. The buttons are color coded to allow for easier identification of the modes by the operator. Below these is another small dial which displays the generator's gain. This dial would not be included on the generator's HMI in the field, but it is useful in testing.

The HMI includes a network interface and communicates with the digital controller over the network using the Modbus/TCP protocol. The HMI sends Modbus read requests to the digital controller once per second to obtain sensor information. The requests consist of the DCS's identifier, the function code, the data address to read, the number of registers or statuses to read, and a cyclic redundancy check (CRC).[16] The DCS responds with a corresponding Modbus response of the same identifier and function code, followed by the number of bytes to follow, the requested information, and CRC. Of the four types of values on the DCS, only two are outputs that can be written to. To modify these, a packet similar to the read requests is sent. This packet also contains the DCS's identifier, function code, the data address to access, and a CRC, as well as the value to write. The function codes used in these exchanges are shown in Table 3.

2.6. Versatility and usefulness of virtualization approach

The testbed is versatile in that the testbed allows changes in sensors, actuators, and other components by using the virtualization framework described by Alves et al. for SCADA testbeds.[2] The configuration file for DCS inputs and output can be modified by the user to specify what general I/O are to be used by the DCS or PLC so that new sensors and actuators may be added (both analog and digital). Also, the simulator, in this case MATLAB, can be modified to add and remove sensors and actuators in the model to correspond with the configuration file. An additional generator can be added to the topology as needed and other changes may be made to the load, etc. Virtual machines that serve as PLCs, HMIs, rogue devices, and other computers may be added as nodes to the virtual network as needed by the user. OpenPLC, which is used in this work as the DCS, does support multiple SCADA protocols, such as Modbus, DNP3, and Ethernet/IP, and also new drivers for protocols to be added. This methodology for virtual testbeds was shown to be representative of an actual testbed as seen in the main works discussing the framework.[2,3] The works discuss a comparison between actual and virtual testbeds, and demonstrate that the testbeds are similar in their response to the same attacks. Therefore, this virtual testbed capability is useful as a way to understand how cyber physical systems would actually behave in the real world and with greater cost effectiveness. As for packets that are sent to various nodes, how they behave depends on the driver for processing Modbus packets. If the same driver is used in the virtual testbed as in the actual testbed, it will respond as expected for a real system. If a malicious rogue device on the network were to send Modbus command packets to the DCS at a high frequency to change the settings of a register in the DCS, a legitimate user may have difficulty also changing the setting while this is happening. The user, who may be able to send command packets to change settings only periodically at a lower frequency from the HMI, would be overruled for the setting by the rogue device. The work by Das and Morris, which uses the virtual modular framework, includes this scenario, demonstrating the framework's versatility in exploring changes of timing in Modbus packets.[17] If a malformed packet is received by the PLC, how the PLC responds depends on the driver implementation for processing packets and also specifically for Modbus. The virtual PLCs in this work use the Modbus implementation of OpenPLC. In this case, if a packet were to be malformed in the application layer of the OSI stack where Modbus resides for Modbus/TCP, then the Modbus packet will not be processed properly and an appropriate error code will be given.[27] This is to be expected in real SCADA systems that use the Modbus protocol. Corruptions of the packets in general, and, more specifically, in the lower layers of the OSI stack would be handled by the standard networking libraries that OpenPLC uses.

3. Experiments with cyber attacks

Diesel generators have been a subject of discussion in cyber-security research for more than a decade, including experiments by Idaho National Laboratory in which the relays connecting a generator to a power grid were hacked.

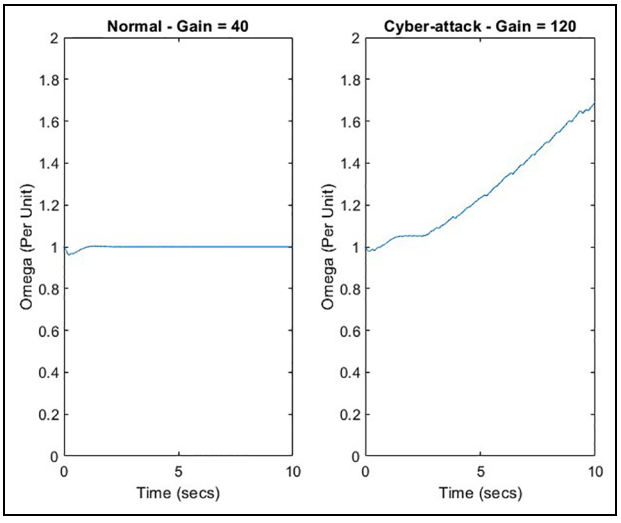

Figure 5. Speed of shaft – normal conditions (left) vs attack on gain (right).

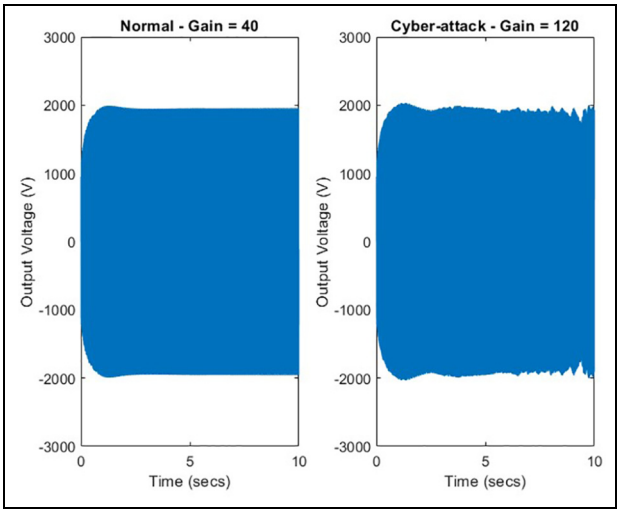

Figure 6. Resulting waveform – normal conditions (left) vs attack on gain (right).

By turning the relays on and off, the generator and grid became unsynchronized, causing physical damage to the generator.[12] This experiment helped bring to light that generators and their associated networks and equipment are vulnerable to cyber attacks.

The purpose of this testbed is to implement additional cyber-physical attacks and observe their effects on the system as a whole. To verify the performance and feasibility of the testbed for these means that examples of two major categories of attacks – network based and supply chain – were simulated. The attacks in these categories were chosen in part because of the adverse effects they can have on networked systems, and in part due to their potential to cause even physical damage.

3.1. Network-based attack scenarios

The following network attacks, in which the attacker is assumed to have access to the ICS network through a rogue device or an unauthorized node, were performed between the DCS and HMI: injection, denial of service, and alteration.

3.1.1. Injection attack. One network-based attack performed was an injection attack against the gain in the control system. The generator's gain is stored as an accessible value on the DCS, so an attacker who has access to the network can inject a single command packet to change the value of the register.[18] During normal conditions, the generator's governor causes the angular velocity (ω, or omega) of the shaft in the generator to converge to the desired setpoint (Figure 5, left). This is necessary to produce output at the correct frequency in a short amount of time and to ensure that the resulting waveform is stable (Figure 6, left).

When the injected packet sends a command to the DCS to change the gain to be three times as large as the original value, the control system is unable to settle on the desired angular velocity. Instead, its behavior becomes chaotic (Figure 5, right) due to its inability to settle upon a constant frequency output. The resulting voltage waveform is irregular in shape, with chaotic frequency output (Figure 6, right), indicating poor power quality. The variations in frequency are great enough to inhibit other connected power sources from remaining synchronized, which could harm the devices being powered by the generator. In industry, a frequency variation of 1.5 Hz is often considered problematic, with variations of 4 Hz or more resulting in damage to attached equipment, even if no other synchronized generators are in use.[19] When the attack occurred, the frequency variation became greater than 30 Hz in 10 s, which far exceeds what is allowed for power systems in the electric grid.

3.1.2. Flooding denial of service. Denial of service attacks were performed using hping3 to send an overwhelming number of packets to the HMI, resulting in the HMI not receiving up-to-date system information from the DCS since the flooding of packets overwhelms the network during the attack. Figure 7 depicts the temperature of the engine versus time as seen from the perspective of the HMI. The temperature is rising as the engine has started. Due to the DoS attack, which occurs between 20 and 40 s into the simulation, new temperature sensor readings are not received by the HMI. This is seen in Figure 7 in the

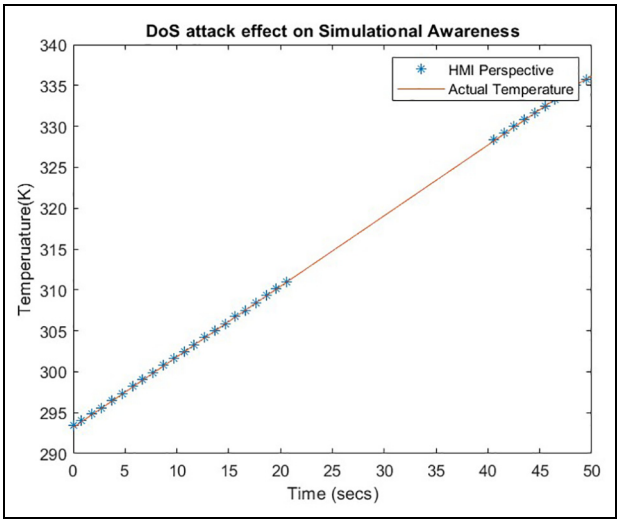

Figure 7. HMI perspective of temperature during DoS attack.
DoS: denial of service; HMI: human machine interface.

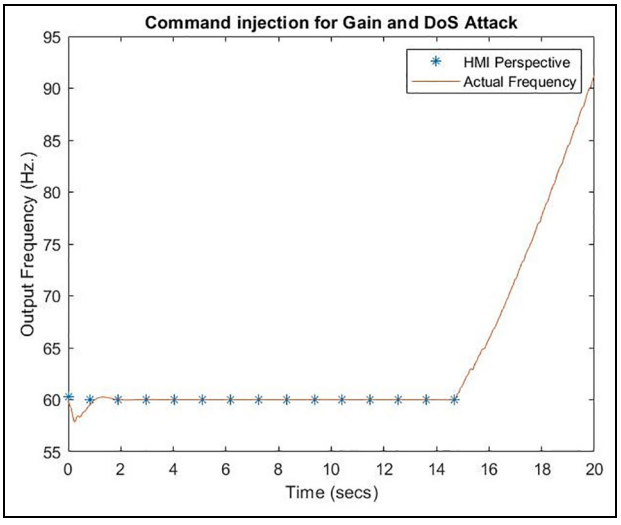

Figure 8. HMI perspective under injection attack on gain and DoS attack.
DoS: denial of service; HMI: human machine interface.

section of the curve without markers indicating no received HMI data, between 20 and 40 s. A DoS attack like this can prevent situational awareness from the HMI's perspective. Therefore, the attack can prevent the user at the HMI from knowing the true situation for critical variables, such as engine temperature. In this case, when the DoS attack ends the temperature readings from the HMI's perspective resume as expected. However, such behavior is not always the case, as can be seen in the next DoS example.

In another scenario of a DoS attack that is more consequential than the previous, an injection attack is performed on the gain parameter of the control algorithm to set the gain to a larger value (Figure 8) immediately before the DoS attack is started. This, in turn, causes instability and chaotic behavior in the engine and the resulting output frequency, which a user would not be able to see. The Attacker's purpose for the DoS attack is to ensure that the HMI, and, therefore, the user, lacks situational awareness and is unable to respond.

3.1.3. Man-in-the-middle alteration. In a Man-in-the-Middle (MitM) alteration attack, packets travelling from their original source are intercepted by an attacker who modifies the packets. The attacker then sends these altered packets to the intended destination while disguising that the packets are not coming directly from the legitimate source. For the network configuration, the HMI and the DCS are assumed to be connected to the same network, specifically the same switch. There is also a rogue device that is connected to the network and is responsible for the MitM attack. To perform this testbed attack, an Ettercap filter was created to read Modbus packets sent from the DCS to the HMI and change the value of the voltage readings in the packet payload. The method by which Ettercap achieves the attack is to use Address Resolution Protocol (ARP) poisoning.

The rogue device conducts ARP poisoning by broadcasting fraudulent ARP packets to victims and changing their ARP table entries. Ordinarily, the ARP protocol allows nodes with IP addresses on the network to discover the MAC addresses of other devices associated with specific IP addresses. This discovery is achieved by broadcasting the request to all other nodes. The node with the corresponding IP address will send its IP address to the requesting node. The requesting node can then put this information in its ARP table. The rogue device takes advantage of the ARP protocol by essentially telling the node that the rogue device's MAC address is associated with a certain IP address. If the node then sends packets to that IP address, the packets will actually be routed and intercepted by the rogue device, which can choose to observe or modify the packets before passing them along to the intended recipient, if at all.[20,21]

Figure 9 plots curves of the generator voltage as seen by the HMI and the actual voltage (sampled from MATLAB Simulink). Prior to the attack, the HMI's display showed the V_{RMS} as 1377 V. This is the correct reading. After the Ettercap attack was started (at approximate time 40 s), the V_{RMS} display in the HMI changes to 1003 V, while the actual voltage is still 1377 V. The attack altered the value as the packets were transmitted, meaning that the operator could only see the modified value of 1003 V and was unaware that the system was not operating at that voltage. This could lead to the operator unnecessarily increasing the output voltage and potentially

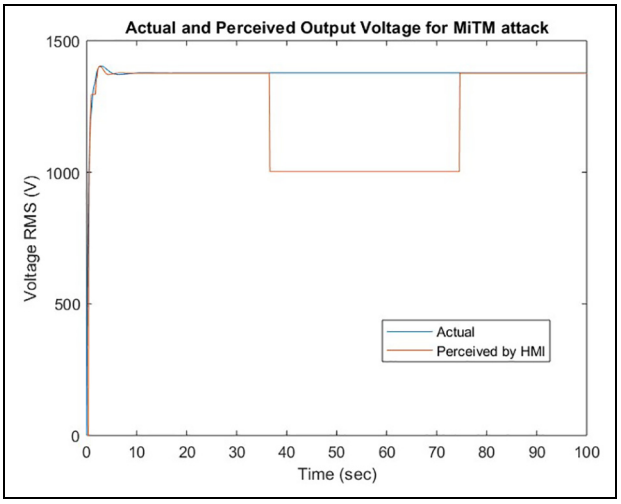

Figure 9. MitM attack affecting voltage reading as seen by HMI. HMI: human machine interface: MitM; Man-in-the-Middle; RMS: root mean square.

damaging connected equipment. This simulated behavior matches the expected behavior of an HMI when subjected to the MITM alteration attack.

3.2. Supply-chain attack scenarios

Supply chain attacks are a result of malicious software or hardware being introduced in the manufacturing process for computers, microcontrollers, and other electronic devices. Two Trojan horse attacks, in which malicious logic is hidden within the device, were performed in these experiments.

3.2.1. Ladder logic Trojan (software based). To emulate a software supply chain attack, the authors implemented a software Trojan in the ladder logic running on the DCS.[22] In this attack, the original ladder logic (Figure 10) was modified to include an additional function block (Figure 11). This software Trojan is triggered after the override is turned on and off five times; this allows for it to not trigger until after the system has been tested and in use for a long enough period of time such that the normal users would consider the equipment to be functioning properly and would therefore trust the equipment. After the fifth set and reset cycle, the output of the Trojan (functionBlock00) is set to zero so that the override will no longer apply.

During normal operation, a system fault, such as low oil pressure, triggers the system to shut down to protect the generator and allow for maintenance. The override is designed to allow the generator to ignore minor faults and remain in operation; it is enacted on generators in critical situations, such as when powering crucial devices at a hospital during a power outage or machinery. With this Trojan enacted, however, the DCS signals the generator to shut down in the case of a minor fault; this attack is modeled after the ladder logic bombs introduced by Govil.[23] This involves subtle changes in the original ladder logic with the goal of causing malicious behavior or impact for applications it was designed to control. Because the attack is not immediately implemented, this method has an advantage in that it is not in effect when the system is first used, making the Trojan harder to detect and perhaps allowing it to bypass normal authentication methods.

3.2.2. Simulated Trojan on sensor (hardware based). A hardware Trojan was also implemented to verify how small changes in the logic of the hardware can have detrimental

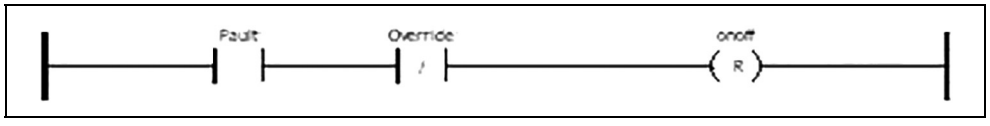

Figure 10. Original override ladder logic.

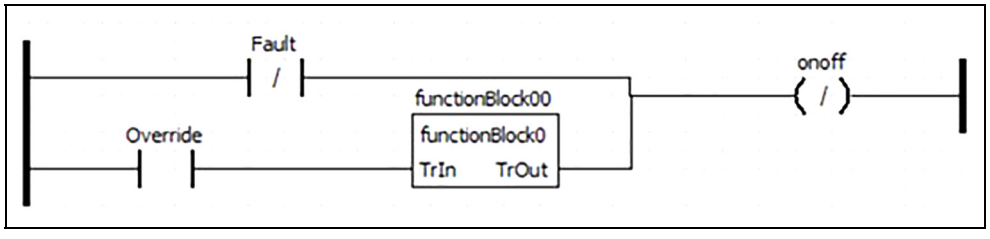

Figure 11. Override ladder logic with software Trojan.

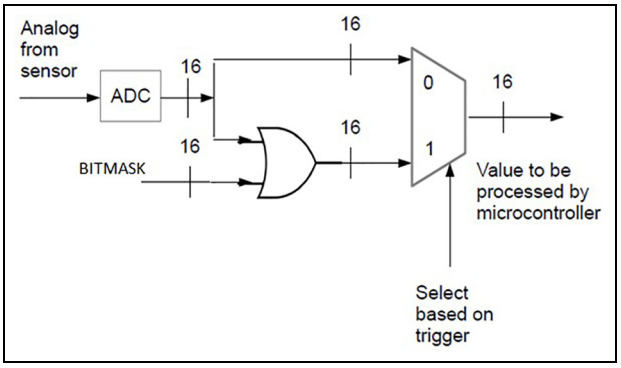

Figure 12. Hardware Trojan affecting temperature sensor reading.
ADC: analog to digital converter.

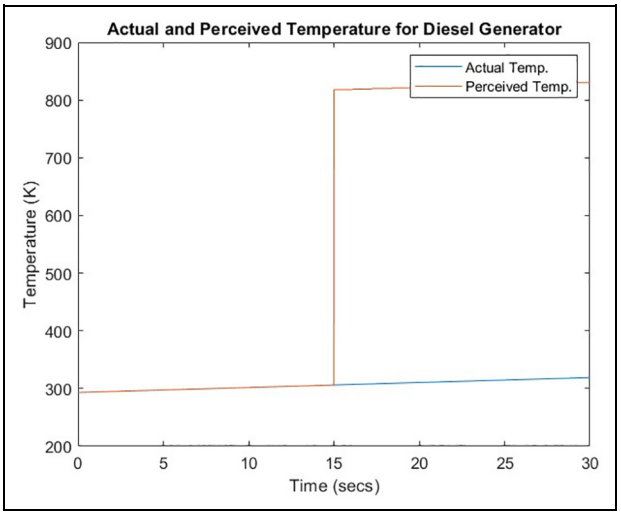

Figure 13. Actual and perceived temperature readings vs. time, with hardware Trojan activated at 15s.

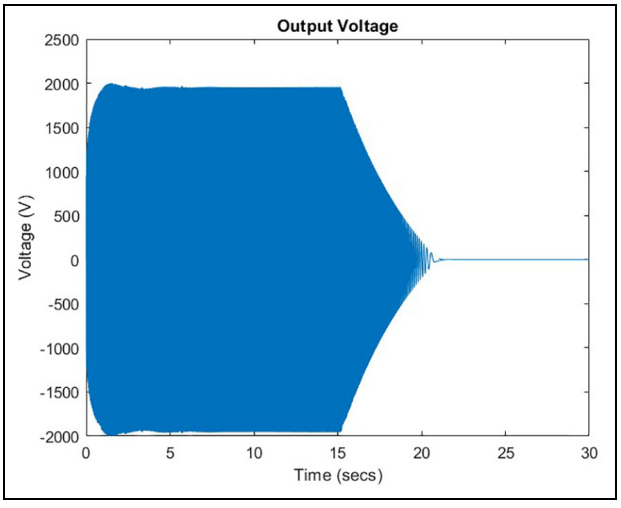

Figure 14. Output voltage vs. time, with hardware Trojan activated at 15 s.

effects on the functionality of the generator system. Under normal operations, analog data from the sensor is passed through an analog to digital converter and then directly to the microcontroller to be processed. This Trojan was inspired by Chakraborty et al. and incorporates a logical OR gate and multiplexer, as seen in Figure 12.[24]

Initially, the sensor data is passed unmodified through the top pathway in the circuitry shown in Figure 12 and into the first input of the multiplexer before being sent to the DCS. Once the Trojan is activated, the sensor data goes through the bottom pathway and is passed through the first input of the OR gate. This gate adds a value to the true sensor reading by performing a binary OR of some of the bits of the sensor data with ones. The most significant bits of the sensor data are usually zero in normal circumstances, so passing them through the OR gate with a 16-bit bitmask (BITMASK), some of whose most significant digits are ones will create a larger output. Because the resulting value read by the DCS is higher than the maximum value in the valid range (300°C or 574 K), a fault is flagged, followed by the automatic shutdown of the generator set. The multiplexer selector can be made to switch the output based upon an external trigger; in this case a timer was used. This hardware Trojan was designed to be activated 15 s into the simulation. This latency is intended to make the Trojan more difficult to detect.

Researchers implemented this Trojan in a temperature sensor on the generator in MATLAB Simulink. The DCS is programmed with a range of safe temperatures for generator operation; if the temperature reading falls outside of this range, the DCS will trigger a fault that will shut down the generator. By artificially inflating the temperature data, this attack causes the DCS to read a temperature that is too high, resulting in an unnecessary system shut down. The temperature value shown in the Simulink model is actually within the safe operating range, but the DCS is observing a fault condition.

Figure 13 shows the correct (actual) and altered (perceived) temperature sensor readings versus time, with a large increase in the perceived value when the Trojan is triggered at 15s. Figure 14 shows the AC voltage over time, with the generator being shut down at 15 s due to the programming of the generator in response to a fault. The fault feature is intended to prevent damage by high temperatures, but when taken advantage of by an attacker, the generator is disabled unnecessarily.

4. Conclusions and future work

Diesel generators are critical to the operation of numerous systems, including providing power to reactor coolant pumps at nuclear power plants and to essential services at hospitals during electrical outages.[10,26] For many decades, the military has used standalone generators. However, more recent military generators are equipped with networking capability. For convenience, it has been desired to be able to remotely monitor the status of generators. Also, because of an interest in micro grids, networked systems are needed for coordination of assets.[25] It is therefore relevant to study the vulnerabilities of generators from a cybersecurity perspective and the impacts of cyber attacks. In contrast to previous works, this work presents a fully virtual high-fidelity testbed composed of a physical model, DCS, remote monitoring system, and the necessary connections. A major contribution of this work is fidelity in its digital control system functions, which includes responding to faults and in its control system algorithm. The integration of other dynamical systems in the testbed, such as the fuel tank and thermodynamic engine model, also serve to improve the overall testbed's fidelity. This testbed provides an opportunity for researchers to implement and observe the effects of cyber attacks without damaging actual, costly systems. The understanding gained from these activities is essential for creating appropriate mitigation strategies to ensure operation of the equipment. Understanding the cyber attacks that may occur with a generator and its associated equipment is important for users, particularly because multiple researchers have demonstrated that cyber attacks can cause physical damage to a generator.

Denial of service, alteration, injection, and hardware and software Trojans were implemented within the testbed and the simulated impact of these attacks matched expected behavior resulting from these attacks when applied against a real system.

The testbed presented in this paper may be used to conduct risk assessments and to develop and evaluate cybersecurity controls and threat mitigation strategies. Additionally, the testbed may be used for cybersecurity training and potential as a honeypot to mitigate against threat actors.

Funding

The material is based upon work supported by the National Science Foundation under Grant DGE-1623657.

ORCID iD

Aaron W Werth https://orcid.org/0000-0003-1936-5627

References

1. Srivastava A, Morris T, Ernster T, et al. Modeling cyber-physical vulnerability of the smart grid with incomplete information. *IEEE Trans Smart Grid* 2013; 4(1): 235–244.
2. Alves T, Das R and Morris T. Virtualization of industrial control system testbeds for cybersecurity. In: *Proceedings of the Second Annual Industrial Control System Security (ICSS) Workshop at the 2016 Annual Computer Security Applications Conference*. Los Angeles, CA. 5–9 December 2016, pp. 10–14.
3. Alves T, Das R, Werth A and Morris T. Virtualization of SCADA testbeds for cybersecurity research: a modular approach. *Comput Security* 2018; 77: 531–546.
4. Korkmaz E, Dolgikh A, Davis M, et al. Industrial control systems security testbed. In: *Proceedings of 11th Annual Symposium on Information Assurance*, Albany, NY, 8–9 June 2016, pp. 13–18.
5. Ashok A, Wang P, Brown M, et al. Experimental evaluation of cyber attacks on automatic generation control using a CPS security testbed. In: *Proceedings of the IEEE Power & Energy Society General Meeting*, Denver, CO, 26–30 July 2015, pp. 1–5. Piscataway, NJ: IEEE.
6. Mallouhi M, Al-Nashif Y, Cox D, et al. A testbed for analyzing security of SCADA control systems (TASSCS). InISGT 2011, Anaheim, CA, 17 January 2011, pp. 1–7. Piscataway, NJ: IEEE.
7. Adhikari U, Morris T and Pan S. WAMS cyber-physical test bed for power system, cybersecurity study, and data mining. *IEEE Trans Smart Grid* 2016; 8(6): 2744–2753.
8. Reaves B and Morris T. An open virtual testbed for industrial control system security research. *Int J Inf Secur* 2012; 11(4): 215–229.
9. Cintuglu MH, Mohammed OA, Akkaya K, et al. A survey on smart grid cyber-physical system testbeds. *IEEE Commun Surv Tutor* 2016; 19(1): 446–464.
10. Yeager KE and Willis JR. Modeling of emergency diesel generators in an 800 megawatt nuclear power plant. *IEEE Trans Energy Convers* 1993; 8(3): 433–441.
11. Gertler J. *Fault detection and diagnosis in engineering systems*. Boca Raton: CRC Press, 1998.
12. Zeller M. Myth or reality – Does the aurora vulnerability pose a risk to my generator? In: *Proceedings of the 64th Annual Conference for Protective Relay Engineers*, College Station, TX, 11–14 April 2011, pp. 130–136. Piscataway, NJ: IEEE.
13. Miller S. Engine cooling model in Simscape. *MATLAB Central File Exchange*, https://www.mathworks.com/matlabcentral/fileexchange/38339-engine-cooling-model-in-simscape (2020, accessed 14 April 2020).
14. Alves TR, Buratto M, De Souza FM, et al. OpenPLC: An open source alternative to automation. In: *Proceedings of the IEEE Global Humanitarian Technology Conference (GHTC 2014)*, San Jose, CA, 10–13 October 2014, pp. 585–589. Piscataway, NJ: IEEE.
15. MathWorks. Convert model from continuous to discrete time – MATLAB c2d, MathWorks Documentation, https://www.mathworks.com/help/control/ref/c2d.html (2018, accessed 5 October 2018).

16. Simply Modbus. Read Coil Status, http://www.simplymodbus.ca/FC01.htm (2017, accessed 14 April 2020).
17. Das R and Morris T. Modeling a midstream oil terminal for cyber security risk evaluation. In: *Proceedings of the International Conference on Critical Infrastructure Protection 2018*, Arlington, VA, 12–14 March 2018, pp. 149–175. Cham: Springer.
18. Morris TH and Gao W. Industrial control system cyber attacks. In: *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013)*, Leicester, UK, 16–17 September 2013, pp. 22–29.
19. Kirby BJ, Dyer J, Martinez C, et al. Frequency control concerns in the North American electric power system. United States Department of Energy. Oak Ridge, TN: Oak Ridge National Laboratory, 2003.
20. Arora H. TCP/IP Attacks – ARP cache poisoning fundamentals explained. Available from: https://www.thegeekstuff.com/2012/01/arp-cache-poisoning/ (2012, accessed 14 April 2020).
21. Wagner R. Address Resolution Protocol Spoofing and Man-in-the-Middle Attacks Practical Assignment GSEC Version 1.2f. Swansea, UK: Sans Institute.
22. Ahmed CM, Mathur A and Ochoa M. NoiSense: Detecting data integrity attacks on sensor measurements using hardware based fingerprints. arXiv preprint arXiv:1712.01598.5 December 2017.
23. Govil N, Agrawal A and Tippenhauer NO. On ladder logic bombs in industrial control systems. In: Katsikas S, et al. (eds) Computer Security. SECPRE 2017, CyberICPS 2017. Lecture Notes in Computer Science, vol 10683. Cham: Springer, pp. 110–126.
24. Chakraborty RS, Narasimhan S and Bhunia S. Hardware Trojan: Threats and emerging solutions. In: *Proceedings of the 2009 IEEE International high level design validation and test workshop*, San Francisco, 4–6 November 2009, pp. 166–171. Piscataway, NJ: IEEE.
25. Marqusee J, Schultz C and Robyn D. Power begins at home: Assured energy for US military bases. Reston, VA: Noblis, 2017.
26. Morgan J. Software optimizes hospital generator, https://www.hfmmagazine.com/articles/3399-software-optimizes-hospital-generator (2018, accessed 14 April 2020).
27. Alves T, Das R and Morris T. Embedding encryption and machine learning intrusion prevention systems on programmable logic controllers. *IEEE Embedded Syst Lett* 2018; 10(3): 99–102.

Author biographies

Aaron W Werth is a PhD candidate studying Computer Engineering at the University of Alabama in Huntsville. He is a recipient of the CyberCorps Scholarship for Service (SFS) and has completed internships at the Tennessee Valley Authority (TVA) and Sandia National Laboratories.

SueAnne N Griffith received her PhD in Computer Engineering at the University of Alabama in Huntsville. She is a former research engineer at the Center for Cybersecurity Research and Education.

Jesse R Hairston is pursuing his PhD in Computer Engineering at the University of Alabama in Huntsville and is a research engineer at the Center for Cybersecurity Research and Education, Huntsville, Alabama, USA.

Thomas H Morris is currently the Eminent Scholar of Computer Engineering, Professor of Electrical and Computer Engineering, and Director of the Center for Cybersecurity Research and Education at the University of Alabama in Huntsville.