A Laboratory-Scale Spillway SCADA System Testbed for Cybersecurity Research

Mohammad E. Alim

University of Alabama in Huntsville, mohammad.alim@uah.edu

Shelton R. Wright

University of Alabama in Huntsville, shelton.wright@uah.edu

Thomas H. Morris

University of Alabama in Huntsville, tommy.morris@uah.edu

This paper describes a testbed that features open-source software and functioning physical processes to model contemporary control systems found in a spillway system for a hydroelectric dam. Didactic descriptions of the testbed intend to provide researchers a replicable tool in cybersecurity in critical infrastructure. Captured network telemetry metrics in addition to network traffic in data logs provide potential insight for training adaptable machine learning models. Use cases in attack scenarios, data logging, and vulnerability assessment illustrate the pragmatic utility of a high-fidelity cyber testbed.

System Security • Network security • Human and societal aspects of security and privacy

Additional Keywords and Phrases: SCADA, testbed, cybersecurity, OpenPLC

ACM Reference Format:

1 INTRODUCTION

Critical infrastructures are systems that are essential for modern society, such that their absence would cause societal functions to fail. Power grids, water towers, gas pipelines, and chemical plants are all examples of critical infrastructures. Supervisory Control and Data Acquisition (SCADA) systems control and monitor critical infrastructures and similar legacy systems. A malfunctioning SCADA system, caused by either equipment failure or by a malicious agent's successful cyber attack, creates a plethora of disastrous consequences for the surrounding populace. Well-known security attacks such as the Maroochy Incident [1], Stuxnet, Duqu [2], and the recent DarkSide ransomware attack [3] are all attacks which targeted critical SCADA systems.

This paper describes a reproducible physical spillway testbed model that closely emulates the operation of a level control system of a hydroelectric dam. Two reservoirs store the water within the testbed and allow a flow of water from an upper primary reservoir to a lower secondary through three solenoid valves with flow sensors interfaced to a programmable logic controller (PLC) for controlling the flow of water from the primary reservoir.

The spillway model incorporates a water level sensor, a set of LEDs, three solenoid valves and corresponding flow rate sensors, one LCD screen, and an alarm. The testbed features open-source software and functioning physical processes to model contemporary control systems found in critical infrastructure.

This paper follows this structure: Section 2 discusses the related literature. Section 3 describes the necessary considerations for designing a generic model of a representative hydroelectric dam and spillway. Section 4 depicts the SCADA system architecture and methods utilized in the approach. Section 5 presents several demonstrations of use cases including attacks conducted against the testbed, data logs collected during the scenarios, and vulnerability assessments performed on the system. Section 6 concludes with final remarks on the technical aspect with consideration for due acknowledgments in Section 7.

2 RELATED WORKS

Cybersecurity constantly evolves to resolve and mitigate novel adversarial attacks which themselves adapt to bypass security mechanisms that are in place. It logically follows that the development and subsequent use of advanced security mechanisms serve to detect, identify, prevent and resolve new contemporary attacks. Researchers develop testbeds to precisely and accurately emulate SCADA systems in distinct industrial environments. Researchers construct virtual testbeds using modeling methods often to avoid the cost associated with constructing physical testbeds that imitate real-world systems. Hybrid testbeds represent a system via scaled physical construction in addition to a virtualized model to provide high-fidelity emulation of real-world applications.

Texeira et al. [4] created a testbed to model a water storage tank's control system and conducted cyber attacks against the model in order to compare several machine learning algorithms in detecting attacks. They remarked that intrusion detection systems (IDS) that use machine learning models trained with a dataset generated at the process control level are more efficient and less complex than traditional protection methods; hence the generation of datasets has become a notable source of interest to researchers. In [5], the authors integrate laboratory-scale control systems into a testbed that features a raised water tower, gas pipeline, factory conveyor, and water storage tank. The work favors the idea of teaching cybersecurity topics to students in an educational setting through the use of testbed demonstrations to provide a reliable, consistent means to validate protection methods for common critical infrastructure systems.

Gao et al. [6] present a hybrid testbed of an industrial boiler using Matlab/Simulink and replicated devices as part of a representative physical construction. They argue virtual testbeds which have no real-world counterpart are insufficient for research purposes due to the low fidelity that arises from a lack of real components subject to factors outside software. Ghaleb et al. [7] present a generic, lightweight SCADA Security Testbed, or SCADA-SST, that uses water distribution and an electrical power grid; each use case has optional physical and simulated components. The goals of this work are to demonstrate an appropriate assessment of resilience of nodes in a SCADA network.

The testbed described herein is a physical tabletop system of similar size to a laboratory-scale control system and capable of thorough data collection of various metrics including network telemetry, metrics, and operation class labels. This system is intended for the research purposes of students in SCADA as well as researchers searching for a reproducible model representative of a real-world system.

3 SPILLWAY MODELING

As pumped-storage hydropower continues to grow as the largest contributor to grid energy storage in the United States with approximately 93% of all commercial storage capacity, a generic commercial dam includes pumped-storage reservoirs and spillways to transport water downstream [8, 9]. The testbed design depicts a generic dam model, typical spillways, and a conventional means to convey sequestered water. As seen in Figure 1, the testbed features a primary water reservoir and a secondary reservoir in a closed-loop pumped-storage hydropower scenario, where water is pumped to a reservoir at a higher elevation when it is needed and discharged to lower elevations by gravity. The testbed does not contain rotating turbines and does not generate power. The secondary reservoir is located inside a cabinet under the testbed and allows the water in the model to be recycled and reused by pumping water into the primary reservoir. The pump fills the primary reservoir to a setpoint water level; the typical operations of the testbed let water flow through the valves when a maximum height is reached.

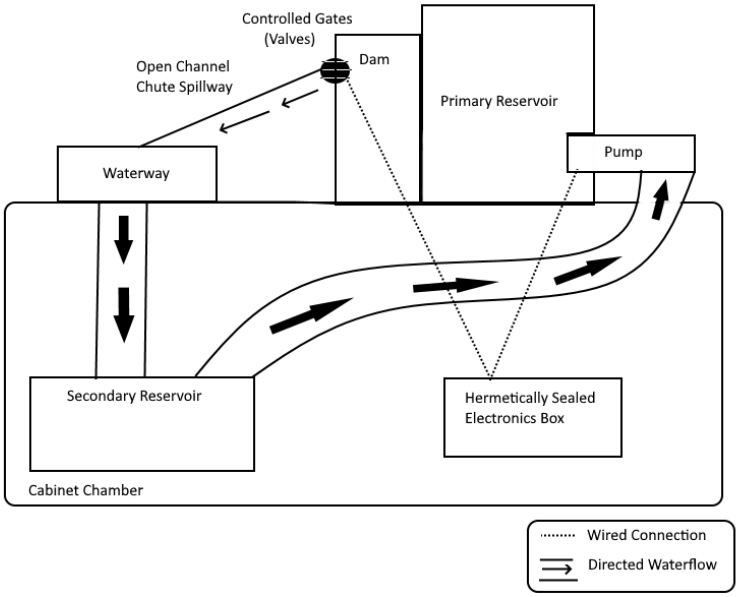


Figure 1: Design of Spillway Model

A spillway is a structure constructed for disposing of surplus water from an upstream region to a downstream region on the other side of the dam site, or to allow a way to control the flow and conveyance from a reservoir to tailwater for all flood discharges up to the spillway design flood [10]. Open channel spillways use the free surfaces of open-channel flow to convey sequestered water from a dam's primary reservoir to a secondary reservoir downstream. The testbed implements a chute spillway instead of a stepped spillway to avoid flow bulking and effects of flow depth that rigidly stepped spillways produce as they carry flow through the slopes of open channel spillways [11]. Once the water conveys out of the primary reservoir, the water flows into a stream and a stilling basin collects the water flow such that hydraulic energy dissipates before reaching the waterway. The depicted waterway includes the stream and stilling basin. The water then circulates back to the secondary reservoir and thus creates a closed loop for the water stream.

The testbed has a simple hydraulic design using a gated ogee spillway to allow water transit and avoid the potential of hydraulic jumping. Gated spillways feature various designs of gates that control or stage waterflow [12]. These systems often pose a set of challenges including the maintenance of reliability operations and mitigating the potential for mechanical or electrical failure to cause adverse consequences on the surrounding environment. The testbed uses solenoid valves as controllable gates to the chute spillways. In addition to the physical requirements of flood mitigation and robust gated structures, a controllable gated spillway system must consider the responsibilities of the operator to maintain and aptly manage the routine operations of the system [13]. This consideration raises the security concern regarding the consequences of a potentially compromised operator during the ongoing operation within critical infrastructures.

4 SPILLWAY SYSTEM TESTBED

Any SCADA system can be logically divided into five basic components. As seen in the top row of Figure 2, these components include the physical system, the cyber-physical link, the distributed control system (DCS), the network connection, and the remote monitoring and control system [14].

The testbed emulates a real-world industrial control system as presented in a hydroelectric dam and avoids implementing a scaled-up operation. A generic model of a hydroelectric dam has controllable gates leading to open channel chute spillways. The components which comprise the model correspond to real SCADA systems and are representative of their respective system processes. The bottom row of Figure 2 depicts the corresponding components of the testbed.

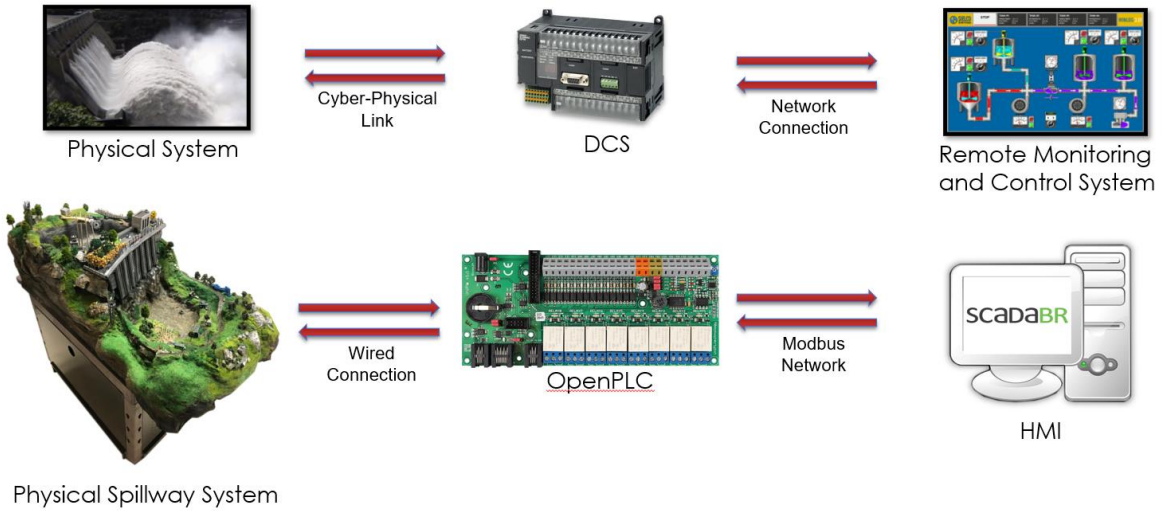


Figure 2: Five Components of SCADA Architecture

4.1 Physical System

The physical system includes sensors and actuators that interface to the cyber portion of the system. There are four sensors in the testbed: the reservoir water level sensor and three water flow sensors. The reservoir water level sensor is built from a tape sensor that outputs an analog voltage that is proportional to the water level in

the reservoir. Three water flow sensors measure the flow rate of water from the upper reservoir to the lower reservoir. Each water flow sensor connects to the output of a solenoid valve. The valves allow for flow control and draining from the upper to lower reservoir. A piezoelectric buzzer on the testbed represents an alarm system, allowing for audible alerts. Three RGB LEDs present a visual alert, staying green while no alarm is set and turning red when an alarm state is entered.

4.2 Cyber-Physical Link

The cyber-physical link provides a transport medium for signals from sensors and actuators in the physical system to reach the PLC. The cyber-physical link for the testbed is primarily composed of electrical wires that transport voltage or current signals to the PLC. Wires from the sensors and actuators connect directly to the I/O terminal of the PLC component allowing the PLC to receive data from sensors and send data to the actuators.

4.3 Distributed Control System (DCS)

The distributed control system (DCS) operates in between the human operator and the physical system. One or more edge controllers often comprise a DCS to provide a direct interface with the sensors and actuators through the cyber-physical link. The PLC is an intelligent industrial computer that monitors sensor data and actuates connected devices according to the programmed ladder logic or the manual control of a human operator. In this testbed model, a Raspberry Pi hosts the OpenPLC Runtime environment [15] and connects to a UniPi expansion board. The UniPi's relays control the alarms, valves, and pump while reading the level sensor data over its analog input ports. The testbed uses an analog input for each sensor. As a UniPi has only two analog inputs, an Arduino Uno supplies additional analog inputs as a slave device to the Raspberry Pi. An Arduino Mega 2560 receives pulse-width modulation (PWM) signals from each of the sensors, determines the corresponding sensor readings, and outputs analog signals to the UniPi for the water level sensor and to the Uno for the flow rate sensors. The Mega collects data from these sensors because the UniPi's OpenPLC system polls the sensors too slowly to produce accurate readings from the PWM signals. The analog outputs for the flow rate sensors are wired through a low-pass filter circuit to convert the PWM output into an analog signal that can be read on the Uno's analog input port. The boards run custom programs written in the Arduino programming language. The Raspberry Pi, UniPi, Uno, and Mega boards are sealed in a hermetic electronics box in the cabinet chamber away from the primary and secondary reservoirs.

The testbed uses custom function blocks in its ladder logic to convert analog readings from the water level sensor to a distance in centimeters, convert an analog reading from a flow sensor to a flow rate in liters per minute, and actively average readings from any sensor to provide a more stable value for a given duration of operation. The maximum height assumed in the testbed is at 10 centimeters. Each flow sensor captures its analog data from an input register variable, which it then passes to the custom function block so that the resulting value is stored in a variable that the HMI may read from. Each actuator connected to a relay on the UniPi has a power rail in the ladder logic so that each controls when the relay opens and closes based on the state of set variables. The PLC has safety logic in place that shuts off the solenoid valves if they overheat from extended use.

The testbed implements an automatic pump control mechanism to determine when to pump water into the model, denoted as Automatic Mode. The pump and control logic together model pumps needed for pumped-storage hydroelectric generation. A corresponding level percentage is calculated with the sensed water level. If

the level percentage is less than the setpoint percentage for three seconds and the pump is not on manual controls, the pump turns on for the duration until the setpoint percentage is reached. The pump waits for three seconds such that a brief fluctuation from the water level sensor does not immediately engage the pump.

Automatic alarm control mechanisms were designed and implemented to control two behaviors regarding the water level, similar to a real-world system. An automatic alarm coil is set if the water level percentage is greater than a designated threshold such as halfway at 50%, otherwise it is set to False. When Manual Mode is enabled, the automatic controls are disabled. If the system engages Manual Mode, then the automatic alarm coil is overwritten.

4.4 SCADA Network Connection

The network connection exists to allow communication between the PLC and the supervisory control and remote monitoring systems. The testbed uses the Modbus TCP/IP protocol to provide the PLC the means to communicate the status of physical processes to the HMI because Modbus is the most popular protocol widely used in industrial control systems [16]. Each device on the network has a static IP. Wireshark was used to observe packets as they move across the network. Readouts of a series of packets on port 502 using the Modbus/TCP protocol show queries and responses between the PLC and the HMI.

4.5 Remote Monitoring and Control System

The remote monitoring and control component of SCADA is composed of historians and data loggers that manage the physical process data and provide a mechanism for user interaction and control. A human operator interacts with a graphical user interface or HMI, which communicates with the PLC through a SCADA protocol such as Modbus. The HMI permits operators to interact with observable variables of the SCADA system necessary for monitoring, analyzing, and controlling the automation process. The spillway testbed uses ScadaBR [17], an open-source web-based user interface utility hosted on an Apache Tomcat server, to monitor and control each of the spillway's sensors and actuators. An overview page shows the status of the entire spillway model for the testbed and an interface depicting detailed information of the solenoid valves controlling the outflow of water from the primary chamber of the spillway. A secondary interface displays information of the current flow of solenoid valves and outflow control from the upper reservoir to the lower reservoir.

5 DEMONSTRATION OF USE CASES

This section presents use cases of the testbed for academic and research purposes. Three separate use cases for the testbed are demonstrated: attack scenarios, data logging, and vulnerability assessments.

5.1 Attack Scenarios

Attacks on spillway components can be demonstrated and evaluated on the scaled testbed model. An attack application was developed, providing a user of the testbed a method to easily configure and launch various attacks. Additionally, an attack script was created to automate the launching of attack scenarios. Automating the attacks from a script instead of the graphical application described above allows for automated running of scenarios to collect large amounts of data with the data logging abilities described in Section 5.2.

The attack application contains a graphical interface that allows a user to configure the IP address and port number of the intended target. Attacks are launched by simply clicking buttons in the graphical window. The

application was written in Python 3 and extensively used the PyModbusTCP library [18] while making calls to traditional offensive cyber tools. Modbus injection attacks were implemented to open and close the valves, turn the pump on and off, and sound or disable the alarm. A denial of service (DoS) attack was implemented by performing a SYN flood on the specified target. The DoS attacks follow three main scenarios based on the service it denies between the PLC, the HMI, and the OpenPLC Runtime web server. When the PLC is attacked, the OpenPLC Runtime crashes and experimenters will need to manually restart it. The HMI is unable to monitor or control the spillway until the PLC has been restarted. When the HMI is attacked, the operator is unable to monitor or control any sensors or actuators. Though, the HMI can resume operations after an attack stops. When the OpenPLC Runtime web server is attacked, the operator is unable to start or stop the PLC Runtime. When the attack finishes, the web server comes back online on its own after a few minutes of idling.

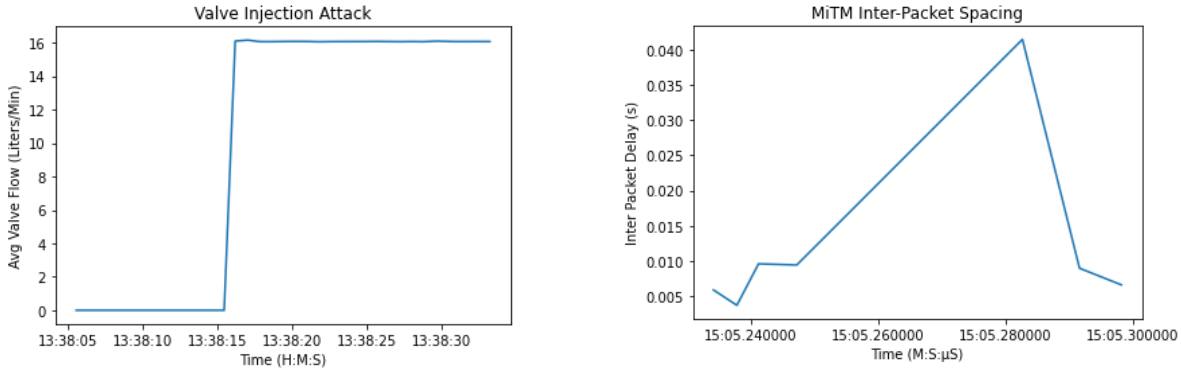
To automate different attack scenarios, an attack script was created to perform various types of cyber attacks. These included reconnaissance attacks, injection attacks against the PLC to change actuator states, and denial of service (DoS) attacks against the PLC itself, the OpenPLC runtime web server, or the HMI. Table 1 lists the attacks carried out on the testbed along with a brief description of the attack.

Reconnaissance attacks provide an adversary with useful information about the system, in this case the spillway testbed. The Device Code Scan sniffs the network traffic and logs all IP and MAC addresses found on the network. The Address Scan looks for the Modbus port 502 on all network traffic and logs the request and response registers found in the network data. Similar to Address Scan, the Function Code Scan monitors Modbus traffic and logs all Modbus function codes found on the network.

The injection attacks inject erroneous Modbus packets into PLC registers and coils. An adversarial agent sending an injection to turn the pump system on can possibly cause flooding in a region if the water level is too high. Disabling the pump system while the water level is low may allow all the water to drain from the upper reservoir, leaving it empty for any future use. An injection attack turning the alarm system or buzzer on may cause an operator to be misled into thinking there is a problem when the water level is actually at a safe height. An adversary injection Modbus traffic to open all spillway valves will allow water to drain from the reservoir when not intended. Figure 3(a) shows the effects of the Injection Open All Valves attack by monitoring the flow through each valve.

Table 1: Testbed Attacks

Attack	**Description**
Recon. Device Code Scan	Network sniffer logging all IP and MAC addresses found on the network
Recon. Address Scan	Network sniffer logging values of Modbus registers found on the network
Recon. Function Code Scan	Network sniffer logging function codes in Modbus traffic
Injection Pump ON	Modbus injection writing pump coil in PLC to "1"
Injection Pump OFF	Modbus injection writing pump coil in PLC to "0"
Injection Buzzer ON	Modbus injection writing buzzer coil in PLC to "1"
Injection Open All Valves	Modbus injection writing all valve coils in PLC to "1"
MiTM/DoS	ARP poisoning or target flooding



(a) Injection Open All Valves: Valve Flow

(b) Man-in-the-Middle: Inter-Packet Spacing

Figure 3: Attack Scenario Results in Data Logger

Man-in-the-Middle (MitM) attacks and DoS attacks were combined into a single category. On the testbed, MiTM consisted of ARP poisoning, allowing an adversary to receive all traffic between the PLC and HMI. This attack is evident when monitoring inter-packet spacing on the receiving end as seen in Figure 3(b).

5.2 Data Logging

A data logger collects network traffic, labels scenarios as normal operations or potential attacks described in Table 1, and calculates network telemetry metrics in a format that is extendible by researchers. The logger generates two comma-separated variable files for Modbus traffic and the overall network traffic, and stores a separate packet capture file for consistent observation of the network traffic.

Data logs include the contents of network traffic packets and telemetry metrics. The collected metrics consist of packet size in bytes, timestamp of packet transmission, inter-packet arrival time, packet process time, protocol overhead and efficiency, throughput, and client flow for the respective client IP. Inter-packet arrival time is the time between packet arrivals at the destination, whereas the packet process time is the amount of time between receiving and transmitting a packet at the destination. These time aggregates show the acceptable rate at which a system operates, which is essential to IDSs and potential attacks that inject packets based on the timing of the system [19]. The protocol overhead is the amount of non-data bits required in transmission of a given packet. The efficiency is defined as the ratio of the data bits to the total bits in a transmitted packet. Throughput is the total number of transmitted bits divided by the time from the first packet to the last packet. Client flow is the sequence of packets from a source address to a destination in a minute time window, which is useful in IP flow-based intrusion detection methods [20, 21]. These metrics were chosen to provide model-based IDS with capture characteristics, packet attributes, and timing-related features.

The generation of such datasets and the extendibility to this framework provide researchers with a tool to train and test IDS and IPS applications [21]. As researchers may use a subset of these parameters, the utility of collecting timestamp, queries, and packet metrics remains relevant for training profilers; for instance, Bernieri et al. recorded timestamp, total Modbus packets, total number of packets, read input register queries, and read input register responses in their network profile generator to determine normal network behavior [22].

5.3 Vulnerability Assessments

The testbed provides a platform to perform vulnerability assessments on real hardware while not requiring access to actual critical infrastructure. This analysis provides important information on vulnerabilities that are possibly found on hardware devices and applications running on the hardware. OpenVAS [23] and Nessus [24] were used to scan the testbed devices for vulnerabilities. These common vulnerability scanners detect potential attack surfaces, provide reports on recorded exploits, and summarize general security for a given service on a node. Combined results from the vulnerability scans are found in Table 2 along with the respective vulnerability classification from each tool. Nessus classifies vulnerabilities on four levels depending on criticality: Critical, High, Medium, and Low. OpenVAS classifies vulnerabilities into three levels: High, Medium, and Low. Vulnerability testing identified a potential Ghostcat vulnerability in the HMI's Tomcat server and possible revelation of host uptime via calculable observations in TCP timestamps in the PLC.

Table 2: Vulnerability Scan Results

Application	Vulnerability	Nessus	OpenVAS
ScadaBR	Apache Tomcat AJP Connector Request Injection (Ghostcat)	Critical	High
ScadaBR	Unsupported Web Server Detection	Critical	-
ScadaBR	CGI Generic SQL Injection (blind)	High	-
ScadaBR	Apache JServ Protocol (AJP) Public WAN (Internet) Accessible	-	High
ScadaBR	SSH Brute Force Logins With Default Credentials Reporting	-	High
ScadaBR	Apache Tomcat Default Files	Medium	Medium
ScadaBR	CGI Generic XSS (persistent, 3rd Pass)	Medium	-
ScadaBR	Web Application Potentially Vulnerable to Clickjacking	Medium	-
ScadaBR	Web Server Uses Basic Authentication Without HTTPS	Low	Medium
ScadaBR	TCP timestamps	-	Low
OpenPLC	Python Unsupported Version Detection	Critical	-
OpenPLC	Web Application Potentially Vulnerable to Clickjacking	Medium	-
OpenPLC	Web Server Transmits Cleartext Credentials	Low	-
OpenPLC	TCP timestamps	-	Low

6 CONCLUSION

This paper contributes a reproducible method of constructing a physical spillway testbed model with accompanying detailed descriptions of the underlying DCS and remote monitoring and control system. The designed and implemented testbed aligns with the research interests in academia and industry. In constructing a testbed, students and researchers can learn from a testbed representative of real-world systems with fidelic cyber components to emulate an authentic system that is cheaper and easier to use than the corresponding existing counterparts. The testbed provides a potential source of dataset generation for research in IDS, IPS, and related cybersecurity development for applications in critical infrastructure.

ACKNOWLEDGMENTS

This work was supported by the U.S. Army Engineer Research and Development Center (ERDC) – W914HZ-16-BAA-02. The authors appreciate the insight provided by Dr. Rishabh Das, Raphael B. Oliveira, Jack Smalligan, and Christopher H. Nguyen.

REFERENCES

[1] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. 2012. The cousins of Stuxnet: Duqu, Flame, and Gauss. Futur. Internet 4, 4 (2012), 971–1003. DOI:https://doi.org/10.3390/fi4040971

[2] Marshal Abrams and Joe Weiss. 2008. Malicious Control System Cyber Security Attack Case Study – Maroochy Water Services, Australia. MITRE Corp USA, August (2008), 73–82. Retrieved from https://www.mitre.org/sites/default/files/pdf/08_1145.pdf

[3] Snir Ben Shimol. 2021. “Return of the Darkside: Analysis of a Large-Scale Data Theft Campaign” Inside Out Security, Varonis. Retrieved September 16, 2021 from https://www.varonis.com/blog/darkside-ransomware/

[4] Marcio Andrey Teixeira, Tara Salman, Maede Zolanvari, Raj Jain, Nader Meskin, and Mohammed Samaka. 2018. SCADA system testbed for cybersecurity research using machine learning approach. Futur. Internet 10, 8 (2018). DOI:https://doi.org/10.3390/fi10080076

[5] Thomas Morris, Anurag Srivastava, Bradley Reaves, Wei Gao, Kalyan Pavurapu, and Ram Reddi. 2011. A control system testbed to validate critical infrastructure protection concepts. Int. J. Crit. Infrastruct. Prot. 4, 2 (2011), 88–103. DOI:https://doi.org/10.1016/j.ijcip.2011.06.005

[6] Haihui Gao, Yong Peng, Zhonghua Dai, Ting Wang, Xuefeng Han, and Hanjing Li. 2014. An industrial control system testbed based on emulation, physical devices and simulation. IFIP Adv. Inf. Commun. Technol. 441, (2014), 79–91. DOI:https://doi.org/10.1007/978-3-662-45355-1_6

[7] Asem Ghaleb, Sami Zhioua, and Ahmad Almulhem. 2017. SCADA-SST: A SCADA security testbed. 2016 World Congr. Ind. Control Syst. Secur. WCICSS 2016 (2017), 34–39. DOI:https://doi.org/10.1109/WCICSS.2016.7882610

[8] Rocio Uria-Martinez, Megan Johnson, and Shan Rui. 2021. 2021 Hydropower Market Report. January (2021). Retrieved from https://www.energy.gov/sites/prod/files/2018/04/f51/Hydropower%20Market%20Report.pdf

[9] U.S. Department of Energy. 2016. A New Chapter for America’s Renewable Electricity Source. (2016), 407. Retrieved from https://www.energy.gov/sites/prod/files/2018/02/f49/Hydropower-Vision-021518.pdf

[10] USACE. 1992. Hydraulic Design of Spillways. Em 1110-2-1603 EM 1110-2-1603 (1992), 170. Retrieved from https://www.publications.usace.army.mil/Portals/76/Publications/EngineerManuals/EM_1110-2-1603.pdf

[11] Hossein Shahheydari, Ehsan Jafari Nodoshan, Reza Barati, and Mehdi Azhdary Moghadam. 2015. Discharge coefficient and energy dissipation over stepped spillway under skimming flow regime. KSCE J. Civ. Eng. 19, 4 (2015), 1174–1182. DOI:https://doi.org/10.1007/s12205-013-0749-3

[12] Design Standards No, General Spillway, and Design Considerations. 2014. Appurtenant Structures for Dams (Spillways and Outlet Works) Design Standard. 14 (2014).

[13] G. S. Paxson, M. W. McCann, and M. E. Landis. 2016. A risk based framework for evaluating gated spillway operations. 6th Int. Symp. Hydraul. Struct. Hydraul. Struct. Water Syst. Manag. ISHS 2016 3730628160, (2016), 613–623. DOI:https://doi.org/10.15142/T3730628160853

[14] Thiago Alves, Rishabh Das, Aaron Werth, and Thomas Morris. 2018. Virtualization of SCADA testbeds for cybersecurity research: A modular approach. Comput. Secur. 77, July (2018), 531–546. DOI:https://doi.org/10.1016/j.cose.2018.05.002

[15] THE OPENPLC PROJECT. Retrieved May 14, 2021 from https://www.openplcproject.com/

[16] Modbus TCP/IP. Retrieved March 17, 2021 from http://www.modbus.org/tech.php

[17] ScadBR. Retrieved May 31, 2021 from https://www.scadabr.com.br/

[18] pyModbusTCP - PyPI. Retrieved May 31, 2021 from https://pypi.org/project/pyModbusTCP/

[19] Bradley Reaves and Thomas Morris. 2012. An open virtual testbed for industrial control system security research. Int. J. Inf. Secur. 11, 4 (2012), 215–229. DOI:https://doi.org/10.1007/s10207-012-0164-7

[20] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras, and Burkhard Stiller. 2010. An overview of IP flow-based intrusion detection. IEEE Commun. Surv. Tutorials 12, 3 (2010), 343–356. DOI:https://doi.org/10.1109/SURV.2010.032210.00054

[21] Yazan Otoum and Amiya Nayak. 2021. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. J. Netw. Syst. Manag. 29, 3 (2021), 1–26. DOI:https://doi.org/10.1007/s10922-021-09589-6

[22] Giuseppe Bernieri, Federica Pascucci, and Javier Lopez. 2017. Network anomaly detection in critical infrastructure based on mininet network simulator. CEUR Workshop Proc. 1816, (2017), 116–125.

[23] OpenVAS – Open Vulnerability Assessment Scanner. 2021. Greenbone Networks. Retrieved August 19, 2021 from https://www.openvas.org/

[24] The Nessus Family. 2021. Tenable ® - The Cyber Exposure Company. Retrieved August 20, 2021 from https://www.tenable.com/products/nessus