SCADAVT–A Framework for SCADA Security Testbed Based on Virtualization Technology

Abdulmohsen Almalawi, Zahir Tari, Ibrahim Khalil and Adil Fahad
RMIT University School of Computer Science and IT
Melbourne, Australia
Emails:abdul.almalawi@student.rmit.edu.au, zahir.tari@rmit.edu.au
Ibrahim.khalil@rmit.edu.au, adil.alharthi@rmit.edu.au

***Abstract*—Supervisory Control and Data Acquisition (SCADA) systems monitor and control infrastructures and industrial processes such as smart grid power and water distribution systems. Recently, such systems have been attacked, and traditional security solutions have failed to provide an appropriate level of protection. Therefore, it is important to develop security solutions tailored to SCADA systems. However, it is impractical to evaluate such solutions on actual live systems. This paper proposes a SCADA security testbed based on virtualization technology, and introduces a server which is used as a surrogate for water distribution systems. In addition, this paper presents a case study of two malicious attacks to demonstrate how the testbed can easily monitor and control any automatised processes, and also to show how malicious attacks can disrupt supervised processes.**

***Keywords*-SCADA, Security, Simulation, Testbed**

I. Introduction

SCADA (Supervisory Control and Data Acquisition) systems control and monitor public infrastructures such as smart grids and water distribution networks and industrial processes. Therefore, any attack targeting these systems could cause great financial losses and have serious impacts on public safety and the environment. For example, the attack on the sewage treatment system in Maroochy Shire (Australia) is an obvious example of such potential attacks on critical infrastructures [1]. The Stuxnet [2] worm, which is designed to damage nuclear power plants in Iran, is a recent example of such threats.

An evaluation of the security of SCADA systems is important. However, actual SCADA systems cannot be used for such purposes because availability and performance, which are the most important issues, are most likely to be affected when analyzing vulnerabilities, threats and the impact of attacks. To address this problem, real SCADA testbeds such as [3] [4] have been set up for evaluation purposes, but they are costly and beyond the reach of most researchers. Similarly, small real SCADA testbeds [5], [6] have also been set up; however, they are still proprietary and location-constrained. Responding to the aforementioned issues, a number of model-based SCADA testbeds have been proposed [7], [8], [9], [10], [11]. However, these testbeds use several modelling tools to build the essential main components of a SCADA system, and the way in which these are linked makes it a complex process to use each testbed. Therefore, such testbeds are unlikely to target experts in control system security.

In this paper we propose SCADAVT, a framework for building a SCADA model-based testbed that is targeted to security experts of SCADA systems. This testbed is built on the top of the CORE emulator [12], and the essential SCADA components such as protocols, I/O modules and simulators of field devices are integrated by exploiting the plug-in service feature which is available in the CORE emulator. Moreover, we introduce a server that simulates water distribution systems through the use of the dynamic link library (DLL) of EPANET, the well-known modelling tool for simulating water movement and quality behaviour within pressurized pipe networks [13]. In addition, the server can simulate any topology of water network systems and can be manipulated by a custom TCP-based protocol.

In Section II, we introduce SCADA systems and their main components. Section III describes the components of the SCADAVT. Section IV introduces the simulations server of water distribution systems. In Section V, we present the scenario implementation of the proposed testbed. Section VI describes two types of malicious attacks and demonstrates their effects on supervised processes. Finally, we conclude the work in Section VII.

II. SCADA System

SCADA systems are used to provide a centralized monitoring and control system for distributed systems such as electric power generation, water distribution and wastewater collection systems and public transportation systems. A SCADA system can be divided into two levels [14]: (i) Control center which contains high-level components such as SCADA servers or Master Terminal Units (MTU), Human-Machine Interface (HMI) and a historian database. From this level, MTU pulls and logs information gathered from field sites, displays information to the HMI, which allows an operator to monitor and control a supervised system, and may send a control message to a field device based upon collected information. Moreover, at this level,

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

all gathered information is stored in a historian database for auditing and analysis. (ii) Field level which encompasses field devices such as Remote Terminal Units (RTU), Programmable Logic control (PLC), and Intelligent Electronic Device (IED). These devices are deployed in field sites to control and collect measurement data from end devices such as sensors and actuators which in this paper are termed 'process parameters'. The measurement data is compiled and formatted by PLCs and RTUs before they are sent to the control center. Various communication links (e.g traditional telephone and computer network and wireless network) are used to link these two levels [15][16]. For further information on a SCADA system, please refer to [17].

III. SCADAVT

The proposed SCADAVT is built on top of the CORE emulator [12] which provides the communication infrastructure for SCADA components. Prior to choosing the CORE emulator as the basis of the proposed SCADAVT, we conducted a thorough investigation of a number of network simulators such as NS2/NS3 [18], OMNET++ [19], OPNET [20], QualNet [21], and emulators such as, PlanetLab [22], NetBed [23] and MNE [24]. The CORE was chosen because of a number of available features: (i) the friendly interface that is used to build any network topology and set up its configuration without writing any code, (ii) the plug-in service that can be exploited to integrate the essential SCADA components, (iii) the CORE emulator that is based on virtualization technology which generates network behaviour and data that are similar to the ones generated by real systems [10]. (IV) Actual network devices can be connected to the emulated network, and therefore, actual SCADA devices can be tested.

A. Integrating SCADA components into the CORE

Due to the lack of SCADA-specific protocols in the CORE emulator, the widely-used SCADA protocol (e.g. ModBus [25]) is implemented to integrate three essential SCADA components: Modbus/TCP slave and master simulators and Modbus/TCP HMI server. These components are integrated as services in the CORE emulator as follows:

1) Modbus/TCP Simulators of Master/Slave: The modern SCADA systems adapted a master-slave model which is similar to the client-server approach, where the role of the slave model is to listen to any request from the master model. The master model sends control messages to a number of slaves to which a required slave responds according to the control instructions received. Therefore, the integration of the master and slave models are important when setting up SCADA systems. Since the proposed SCADAVT in this stage supports only the Modus protocol, the MODBUS/TCP Master/Slave modes are integrated as shown in Figure 1. The publicly-available ModBus library [26] is used to integrate the simulators in the CORE emulator. This is performed by some python-based scripts that automatically do this integration, where simulators are added to the CORE emulator as services.Therefore, the user does not need to write any code.

Similar to the actual field device (e.g PLC or RTU), each Modbus/TCP slave simulator in the CORE emulator after integration is required to map its registers. Therefore, we provided a registers map procedure for mapping the registers as follows:

```
pro_1 = ['ProcessID1','C',1,'i'];
pro_2 = ['ProcessID2','C',2,'i'];
...
...
pro_9 = ['ProcessID9','C',10,'o'];
Registers=[pro_1,pro_2,..,pro_9];
```

The pro 1 is a python list variable which contains the tag of a supervised process parameter and its block type and position in the RAM in its associated slave simulator and parameter type (e.g, input/output). Four symbols, namely H, C, D and A are used to represent the following register types, HOLDING, COILS, DISCRETE and ANALOG registers respectively. It can be seen from the above that the type of registers are COIL. The last line is the function which adds the mapped registers. All the process IDs have to be unique. This is because the gateway reads and writes measurement data from and to the registers in each Modbus/TCP slave simulator through the ID process.

2) Modbus/TCP Simulator of HMI Server: The HMI Server is an intermediate component between MTU and the HMI client where HMI client sends the user's manipulation to the HMI server in order to be read and executed by the MTU. In the opposite direction, the MTU sends to the HMI server the collected data from a field device after the user's manipulation so that the HMI client can request it in order to show the effects of the user's manipulation in a graphical interface for human operators. As can be seen in Figure 1, the HMI client is considered as an external component in the proposed SCADAVT because the HMI client with a graphical interface cannot be supported in the CORE emulator. Therefore, the simulator of the HMI server runs two independent instances: the first instance listens to the request from the MTU via the internal IP of virtual node in the emulated environment, while the second instance listens to the request from the HMI client via the *backchannel* which is assigned to each emulated node. Therefore, the HMI client can connect with the HMI server in two ways: via a *backchannel* or directly through RJ45 if the HMI client has an independent physical interface and supports Modbus/TCP protocol as well.

3) I/O modules Simulator: The Modbus/TCP slave simulator, which will be running in the virtual node, is required to monitor and control the simulated supervised process such as the simulations of power generation and water distribution

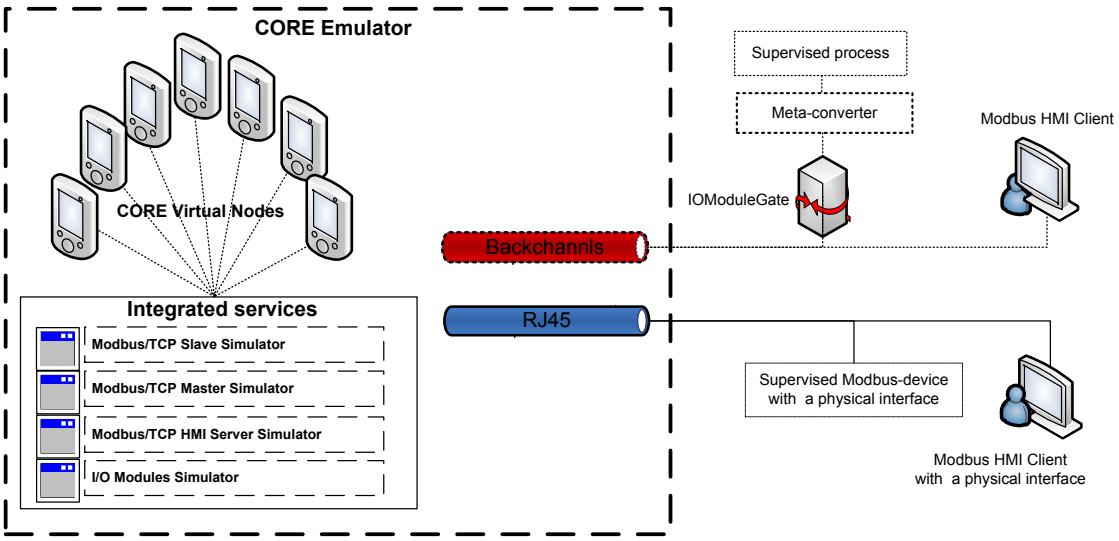

Figure 1: SCADAVT Architecture.

systems that are outside the emulated environment. Therefore, the *IOModules* simulator is integrated into the CORE emulator, where it acts as a server which receives input data from the external environment and sends output data when requested. This is performed through the *backchannel* for each virtual node using a simple and intuitive custom TCP-based protocol called *IOModules* that will be elaborated on in Section III-C.

B. IOModuleGate

A gateway class implements the IOModules protocol (see Section III-C). This gateway class periodically exchanges the measurement data between each slave simulator and the respective supervised process parameters through the I/O modules server running in each virtual node. Two configuration files are invoked to this class, where each file is formatted as shown in Table I. For example, in the first file, PLC1 represents the ID for a slave simulator whose *backchannel* IP and port are 172.16.0.1 and 9161 respectively. While, in the second file, the process parameters P1 and P2 are supervised by this slave simulator (PLC1). The parameter type is indicated by either ”o” or ”i” (e.g, input/output). For pulling and pushing the measurement data to and from the emulated environment, two public writing and reading methods are provided by this class. These methods take and return a python dictionary variable which is a key-value pair. The key is the identity ID of the process parameter (e.g P1) and its I/O data, where each process parameter in a supervised process must have a unique ID.

C. Communication protocols

Two types of protocols are used in SCADAVT: (i) SCADA protocol such as Modbus that is used to enable communication between the SCADA components within the CORE network emulator, and also to communicate with external components (e.g. simulated or real devices) which support the Modbus protocol. (2) A custom TCP-based protocol called IOModules to exchange the measurement data between the I/O modules simulator and the gateway class which called IOModuleGate.

Table I: The configuration of IOModuleGate

1st conguration file	2nd conguration file
[PLC1] ip:172.16.0.1 port:9161 [PLC2] ip:172.16.0.2 port:9161	[P1] controller : PLC1 paraType : i [P2] controller : PLC1 paraType : o

1) SCADA protocol: The Modbus protocol worked only on Modicon programmable controllers. However, it has become widely-used in recent SCADA control devices. Modbus devices adapted a client-server approach, where the Modbus slave device represents the server side, while the Modbus master device represents the client side of the communication model. Only the master (Client) initiates the communication, while the slave (Server) listens to the request from master in order to supply the requested data or execute the requested action. There are many variants of Modbus protocols. In this testbed, we have used the Modbus/TCP protocol. Please refer to [25] for more details about this protocol.

2) IOModules protocol: A simple custom TCP-based protocol is used to read/write the measurement data from

I/O modules simulator that resides in each virtual node. This protocol is implemented by the IOModuleGate class. Figure 2 shows four fields that comprise the message structure of this protocol: (1) *TransactionNo*: a unique number for each reading and writing operation. Both reading and writing operations have independent sequential numbering and initially start with one. In the response message, this field contains the same number of request messages to indicate that output data is available and correctly read. However, if it contains zero, it indicates the output data is not ready to be read, and therefore it needs to wait a while before requesting again. The amount of waiting time can be specified in the initialization time of the IOModuleGate class. (2) *Function Code*: this takes three values, 1, 2 and 0, to indicate reading, writing and termination operations respectively. (3) *ProcessID*: each process parameter in a supervised process must have a unique ID. (4) *Data*: it contains the process parameter's value. This field is set to zero in the request message of the reading operation.

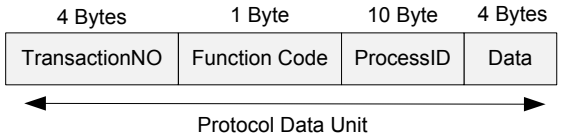

Figure 2: IOModules protocol message structure.

IV. WaterSystem Server

In this section, we introduce a server that simulates water distribution systems using a dynamic link library (DLL) of the well-known modelling tool, which is called EPANET, for simulating water movement and quality behaviour within pressurized pipe networks [13]. This server is designed by Visual Basic 6 langauge. Three arguments are required: (i) the description file that describes the topology and properties of all components for the simulated water distribution system. This description file can be designed and exported by the visual interface of EPANET tool; (ii) port number which the server is listening on; and (iii) time interval to recompute new simulated data.

The server is provided with a custom TCP-based protocol in order to manipulate the simulated data using a SCADA system. For instance, a client can acquire and control pump status in the simulation through this protocol. Figure 3 shows the message structure of this protocol. (1) The *processID* contains the ID of the process parameter, which needs to be manipulated. (2) The *Function code* defines only two operation types: acquisition and control. (3) The *Parameter Type* defines only two component types, *Node* and *link*. (4) The *Data Type* specifies the data type that needs to be manipulated. For example, the link component such as pump has a number of data types that can be manipulated (e.g. speed, pumping energy and status). The (5) *Data* contains the process parameter's value. This field is set to zero in the request message for the reading operation.

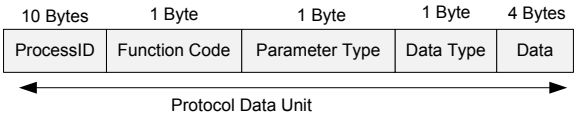

Figure 3: The protocol message structure of the WaterSystem Server.

V. Scenario Implementation

In this section, we demonstrate the implementation of SCADA system that supervises and monitors water distribution network in order to evaluate the proposed testbed. This scenario is performed in three steps as follows:

A. Setting up SCADAVT

To set up SCADAVT, a number of dependencies are required to be installed as follows:

- *The CORE emulator* is the core part of SCADAVT. For installation details, please refer to [12].
- *The Modbus library* is a library provided by a third party and is publicly available [26]
- *Python interpreter* is a prerequisite for the CORE emulator, Modbus library and our integration scripts.
- *the hpin3 utility* is a security assignment tool that is used to assemble/analyze TCP/IP packets [27]. This tool supports many protocols such as TCP, UDP and ICMP. Moreover, this tool can be used to launch a number of attacks such as Denial of Service and spoofing attacks.
- *The integration scripts* are our python scripts developed in order to integrate the essential SCADA components into the CORE emulator as services. To automatically add these services, a user needs to move these scripts to the *myservices* directory which is found in the CORE default directory path prior to staring up the CORE emulator.

B. The setup of water distribution system

In this scenario, we designed a Water Distribution System (WDS) for a small town as shown in Figure 4 using the graphical interface of the EPANET tool. Figure 4 shows that the water network is divided into three areas, namely A, B and C. Each area has an elevated tank to supply the area with water at a satisfactory pressure level. The supplied water is pumped out by three pumps from the treatment system into $Tank_1$. The water is also delivered to $Tank_2$ by two pumps. $Tank_3$ is supplied through gravity because the elevation of $Tank_2$ is higher than $Tank_3$. $Tank_1$ is twice the size of $Tank_2$ and $Tank_3$ because it is the main water source for areas B and C. left-bottom-right-top The water consumption in the water network model is one of the factors that reflects the behaviour of simulated data. Therefore, a

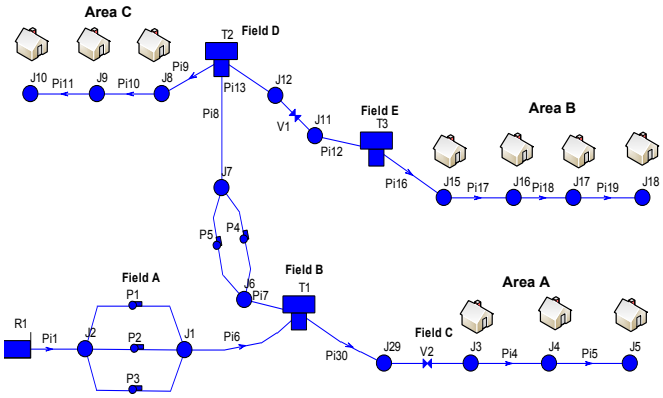

Figure 4: The Simulation of water distribution system

realistic model of water consumption behaviour is required in order to obtain more realistic simulated data. Therefore, we fed the consumption module, in the EPANET model, with a specific model based on [13] (i.e. the 2010 Melbourne water consumption).

C. The setup of SCADA system for WDS

In this section, we demonstrate the deployment of the SCADA devices that are used to monitor and control the previously-discussed water distribution system. This process is performed by dragging and dropping the components of the CORE emulator such as virtual node, link and router. The integrated SCADA components (e.g. Modbus/TCP slave simulator) are automatically added to the services that can be assigned to any virtual node by one click. Figure 5 shows the SCADA network topology for this scenario. Seven PLCs are deployed, and Table II shows their respective supervised devices (sensors or actuators). Each PLC is required to have the services of I/O modules and Modbus/TCP slave simulators. Then, we map its registers to supervised devices that are responsible for it, as discussed in the previous Section III, and assigns its IP address and port. All these PLCs are managed by the Master Terminal Unit (MTU) which is represented by a virtual node that has to have the service of the Modbus/TCP Master Simulator. MTU, in this scenario, performs the following functions:

- Reads the water level in $Tank_1$ from $PLC3$, and then sends control messages to $PLC2$ to turn ON/OFF the actuators of the pumps $Pump1$, $Pump2$ and $Pump3$
- Reads the water level in $Tank_2$ from $PLC1$ and then sends control messages to $PLC4$ to turn ON/OFF the actuators of the pumps $Pump4$, $Pump5$.
- Reduces the water pressure in area A when water level in $Tank_1$ becomes higher.
- Intelligently adjusts the flow valve V_1, which is between $Tank_2$ and $Tank_3$, with an appropriate setting. This is done using Algorithm 1.

Table II: Field devices and their respective supervised devices

Field devices	Supervised devices
PLC1	[T2], water level sensor of $tank_2$
PLC2	[P1, P2, P3], pump actuators of pump1, pump2 and pump3
PLC3	[T1], water level sensor of $tank_1$
PLC4	[P4, P5], pump actuators of pump4 and pump5
PLC5	[V2], valve actuator of valve, V2
PLC6	[V1], valve actuator of valve, V1
PLC7	[T3], water level sensor of $tank_3$

Algorithm 1 A smart control algorithm controlling water flow from $Tank_2$ to $Tank_3$

```
b ⇐ Water demand in area B
c ⇐ Water demand in area C
t2 ⇐ Water level in tank2
t3 ⇐ Water level in tank3
f ⇐ Water flow to tank2
if t2 > t3 then
    flow = b + (f − c)
    Adjust V1 to flow
else
    flow = b − c
    Adjust V1 to flow
end if
```

Algorithm 1 is implemented to maintain sufficient water in both $Tank_2$ and $Tank_3$. This problem is illustrated in Figure 6 where the water level of $Tank_2$ reached the critical level seven times, during which area C was not efficiently supplied with water. This is because the flow valve V_1 is set to a fixed setting, which is 1300 Litre Per Minute (LPM) in this example, and the water flow from $Tank_2$ to $Tank_3$ is constant even though the water level in $Tank_2$ is low. This problem is addressed by considering the following parameters: the water level in $Tank_2$ and $Tank_3$, the current water demands in areas C and B and the water flow pumped in to $Tank_2$. These parameters are used by the MTU to intelligently adjust the flow valve V_1. Figure 7 shows the water level of $Tank_2$ and $Tank_3$ after applying this algorithm. Thanks to SCADA systems, there is an increase in the performance of daily services with less equipment.

To start up the manipulation of the simulated data in WDS, two steps are required: (i) invoking the WaterSystem Server with required arguments that are previously discussed in Section IV; (ii) extending the *IOModuleGate* class and invoking it with configuration files of the system according to the format that was previously discussed in Section III-B.

Clearly, from the detailed discussions above, the functioning of the emulation requires several different configuration steps which necessitate specific knowledge of SCADA systems. Therefore, the user of SCADAVT who performs the simulation has to be someone well-versed in the specifics of

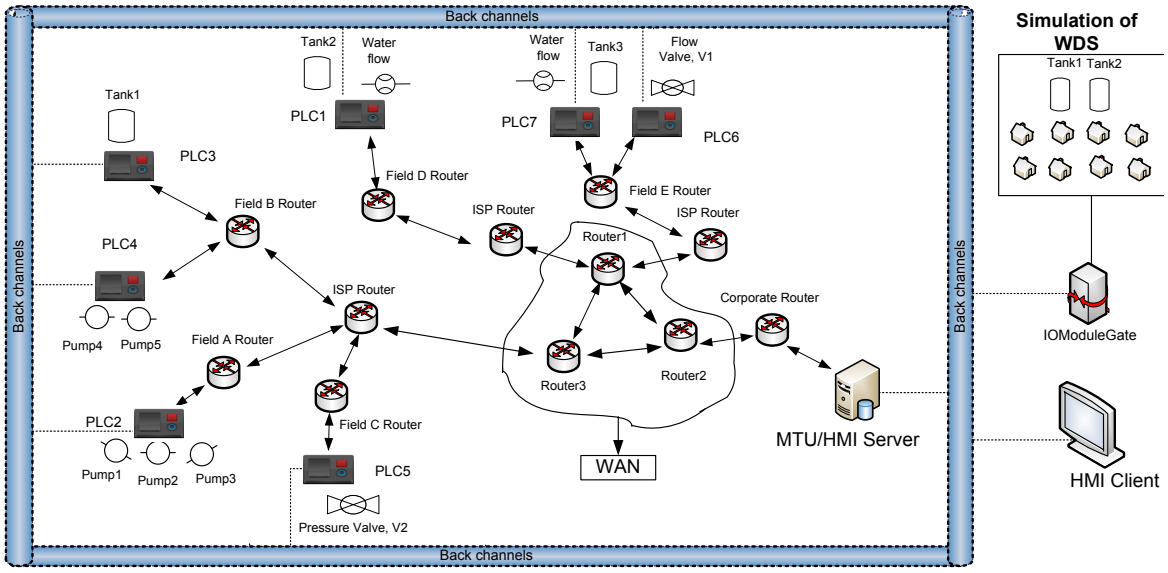

Figure 5: SCADA network topology for controlling the scenario of the water distribution network

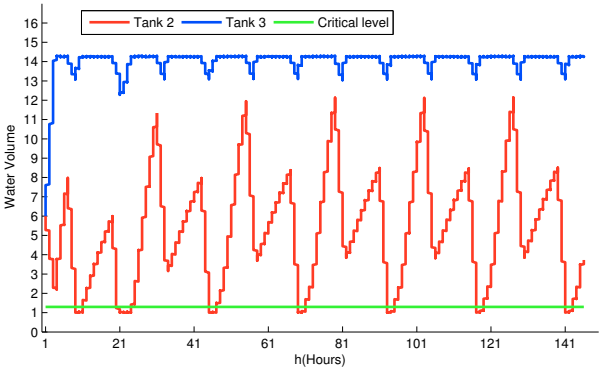

Figure 6: The water levels over a period of time for $Tank_2$ and $Tank_3$ without control system

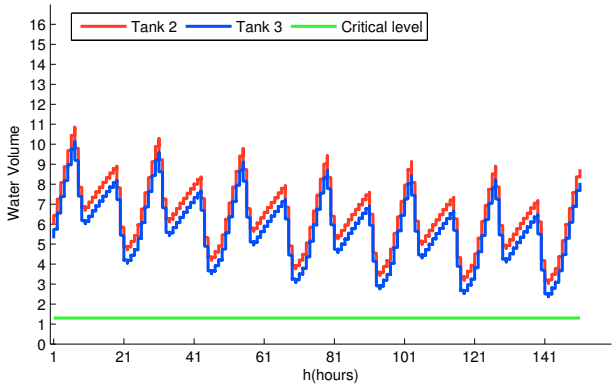

Figure 7: The water levels over a period of time for $Tank_2$ and $Tank_3$ with control system

SCADA systems.

VI. Attack Scenarios

We demonstrate two common attack scenarios: a *denial of service* and an *integrity attack* to evaluate SCADAVT and show how these malicious attacks can affect the performance of the WDS.

A. Denial of Service Attack

In this type of attack, attackers launch flood attacks against a target to prevent it from receiving a legitimate request. As previously discussed, the MTU periodically adjusts the flow valve V_1, which is controlled by PLC_6, using Algorithm 1. In fact, if the MTU cannot establish a connection with PLC_6 to send a control message, the flow valve V_1 will not be properly adjusted. Hence, the water volume in $Tank_2$ and $Tank_3$ will not be balanced, and the critical level may be reached. In this scenario, we demonstrate a Distributed Denial of Service attack (DDoS) where ten virtual nodes are attached to public routers, namely router1, router2 and router3. This is easily done with a few clicks. The open source hping3 utility [27] is used to launch flood attacks on the field device PLC_6. Three times PLC_6 was flooded with TCP SYN packets. The first attack starts at time= 15h and ends at 20h. The second attack starts at time= 55h and ends at 57h. The last attack starts at time= 100h and ends at 105h. During these attack times, the MTU sometimes failed to establish a connection with PLC_6 and sometimes it took a long time to successfully connect with it. Figure 8 shows the unsuccessful and successful connections between MTU and PLC_6. It can be seen that the unsuccessful connections and

the connection establishing time at the period time of DDoS are significantly different from the normal behaviour. That is, the MTU failed a number of times and waited a long time to establish connection compared to attack-free time. Hence, because the MTU failed to intelligently adjust the flow valve V_1, the water volumes of both $Tank_2$ and $Tank_3$ have been affected. Figure 9 clearly shows that the water volumes of $Tank_2$ and $Tank_3$ fell below the critical level twice and once respectively. Consequently, areas C and B were not sufficiently supplied with water two and one times respectively.

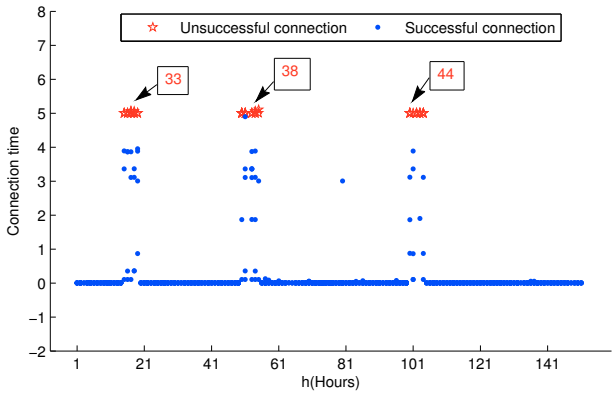

Figure 8: The unsuccessful and successful connections and their elapsed times between MTU and PLC_6

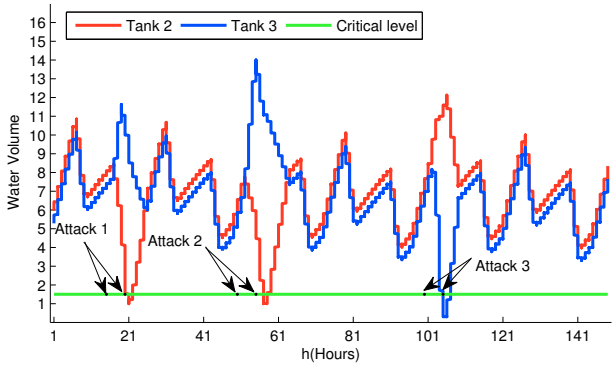

Figure 9: The effect of DDoS, which targets PLC_6, on the water volume of $Tank_2$ and $Tank_3$.

B. Integrity Attack

This type of attack occurs as a result of the manipulation of command messages; it is termed a *high-level control attack* [7] [28] [29]. To launch this type of attack, an attacker requires prior knowledge of the target system. This can be obtained by the specifications, or by a correlation analysis of the network traffic of that system. Taking over the control center and sending undesired control messages, or intercepting (e.g, man-in-middle attack) the command messages between the control center and field devices, are a common means of launching such attacks. In fact, such an attack is difficult to detect because the false message is still legitimate in terms of the Modbus/TCP protocol specifications. To demonstrate this type of attack, we intercept and modify the control message between the MTU and field device PLC_4, which controls the operation of $Pump_4$ and $Pump_5$. To perform this attack, we establish a proxy between these devices. As previously discussed, the MTU sends a control message to PLC_4 to turn its associated pumps ON/OFF. We modified the intercepted control message sent to PLC_4 three times. The starting and ending times of each integrity attack are depicted in Figure 10. In each attack, we modified the intercepted control message with a control data whereby $Pump_4$ and $Pump_5$ are turned off. Figure 10 shows the water volumes of $Tank_2$ and $Tank_3$ after the integrity attack, and it can be seen that the critical level was reached several times, and that the effect of attacks has not occurred at the same times as the attacks. This depends on the functionality being attacked in the target system.

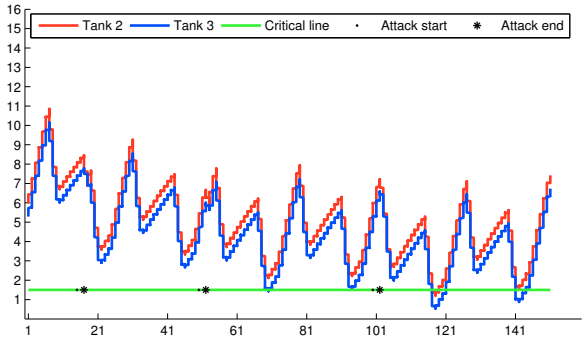

Figure 10: The effect of integrity attack, which targets PLC_4, on the water volume of $Tank_2$ and $Tank_3$.

VII. Conclusion

This paper presents a framework for a SCADA testbed based on virtualization technology. The proposed framework is a novel solution in that, unlike existing testbeds, it provides a friendly interface to create a SCADA system with just a few clicks. Moreover, it is based on the CORE emulator that is scalable for an increasing number of virtual nodes. In addition, the proposed framework realistically mimics the real SCADA testbed, and also has the feature that can allow an actual SCADA device to be connected for realistic evaluation. Furthermore, the paper introduces a server which acts as a surrogate for water distribution systems. A case study is presented to demonstrate how the testbed can easily be used to monitor and control any automatised processes. DDoS and integrity attacks have been described to illustrate how malicious attacks can disrupt

supervised processes. In future work, we intend to integrate other SCADA protocols such as *Zigbee* and *DNP3* .

REFERENCES

[1] J. Slay and M. Miller, "Lessons learned from the maroochy water breach," in *Critical Infrastructure Protection*. Springer Boston, 2007, vol. 253, pp. 73–82.

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32.stuxnet dossier," Symantec Tech, Tech. Rep. 1.4, 2011.

[3] H. Christiansson and E. Luiijf, "Creating a european scada security testbed," in *IFIP International Federation for Information Processing*, vol. 253, no. 1, 2010.

[4] "National scada testbed," July 2012. [Online]. Available: http://www.sandia.gov/ccss/National_Testbed.htm

[5] I. N. Fovino, M. Masera, L. Guidi, and G. Carpi, "An experimental platform for assessing scada vulnerabilities and countermeasures in power plants," in *Human System Interactions (HSI), 2010 3rd Conference on*. IEEE, 2010, pp. 679–686.

[6] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A control system testbed to validate critical infrastructure protection concepts," *International Journal of Critical Infrastructure Protection (IJCIP)*, vol. 4, no. 2, pp. 88–103, 2011.

[7] C. Queiroz, A. Mahmood, and Z. Tari, "Scadasim-a framework for building scada simulations," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 589–597, 2011.

[8] N. Kush, E. Foo, and E. Ahmed, "Smart grid test bed design and implementation," 2010.

[9] C. M. Davis, J. E. Tate, H. Okhravi, C. Grier, T. J. Overbye, , and D. Nicol, "Scada cyber security testbed development," in *Power Symposium, 2006. NAPS 2006. 38th North American*. IEEE, 2006, pp. 483–488.

[10] B. Reaves and T. Morris, "An open virtual testbed for industrial control system security research," *International Journal of Information Security*, pp. 1–15, 2012.

[11] A. Giani, G. Karsai, T. Roosta, A. Shah, B. Sinopoli, and J. Wiley, "A testbed for secure and robust scada systems," *SIGBED Rev*, vol. 5, no. 2, pp. 1–4, 2008.

[12] J. Ahrenholz, "Comparison of core network emulation platforms," in *Proceedings of IEEE MILCOM Conference*, 2010, pp. 864–869.

[13] [Online].Available:http://www.epa.gov/nrmrl/wswrd/dw/epanet.html, "Software that models the hydraulic and water quality behavior of water distribution piping systems," Accessed November 2011.

[14] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST Special Publication*, vol. 800, p. 82, 2007.

[15] A. N. Bessani, P. Sousa, M. Correia, N. F. Neves, and P. Verssimo, "The crutial way of critical infrastructure protection," *Security Privacy, IEEE*, vol. 6, no. 6, pp. 44 –51, Nov.-Dec. 2008.

[16] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *Computers & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[17] D. Bailey and E. Wright, "Practical scada for industry," 2003.

[18] "Ns3 simulator," July 2012. [Online]. Available: http: www.nsnam.org/

[19] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 60.

[20] "Opnet modeler: Scalable network simulation," July 2012. [Online]. Available: http://www.opnet.com/solutions/network_rd/modeler.html

[21] "Scalable network technologies: Qualnet developer," July 2012. [Online]. Available: http://www.scalable-networks.com/products/developer.php

[22] "Planetlab: an open platform for deploying," July 2012. [Online]. Available: http://www.planet-lab.org/

[23] "Emulab-network emulation testbed home," July 2012. [Online]. Available: http://boss.netbed.icics.ubc.ca/

[24] "Mobile network emulator (mne)," July 2012. [Online]. Available: http://cs.itd.nrl.navy.mil/work/proteantools/mne.php

[25] M. IDA, "Modbus messaging on tcp/ip implementation guide v1.0a," June 2004.

[26] "Modbus library," July 2012. [Online]. Available: http://code.google.com/p/pymodbus

[27] "hping3," July 2012. [Online]. Available: http://www.hping.org/hping3.html

[28] D. Wei, Y. Lu, M. Jafari, P. M. Skare, and K. Rohde, "Protecting smart grid automation systems against cyberattacks," *IEEE Transactions on Smart Grid*, no. 99, pp. 1–1, 2011.

[29] A. Giani, E. Bitar, M. Garcia, M. McQueen, P. Khargonekar, and K. Poolla, "Smart grid data integrity attacks," no. 1, January 2012.