ORNL/TM-2023/3120

Virtualizing Industrial Control Networks for Cyber Resilience Experiments

Jordan Johnson
Ali Hanson
Danica Hahn
Jennifer Guerra
Aaron Werth
Andrew Herron

August 2023

OAK RIDGE National Laboratory

ORNL/TM-2023/3120

Cyber Resilience and Intelligence Division

VIRTUALIZING INDUSTRIAL CONTROL NETWORKS FOR CYBER RESILIENCE EXPERIMENTS

Jordan Johnson
Ali Hanson
Danica Hahn
Jennifer Guerra
Aaron Werth
Andrew Herron

August 2023

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831
managed by
UT-BATTELLE LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Table of Contents

CONTENTS iii
ABSTRACT 4
1. Introduction 4
    1.1 Related Works 4
    1.2 Organization of Report 5
2. Motivation 5
3. SCADA 5
    3.1 Mango OS for SCADA Master 5
    3.2 Mango Configuration Tool 6
4. Virtual Network 6
5. Virtual Devices 7
6. Case Studies 7
    6.1 Complex Network Topology Use Case 8
    6.2 Hybrid Virtual/Actual hardware Use Case 10
7. Limitations 14
8. Conclusions 14
References 14

ABSTRACT

Industrial control systems (ICS) networks are undergoing constant shifts to accommodate new security measures. It is challenging to test varying network configurations and security tools with physical systems as they typically include large, expensive equipment. Not only this, but researchers often do not have access to this type of equipment for development of new security tools and techniques. As a solution to these issues, this work presents a set of tools for utilizing GNS3 and Docker as a virtual ICS network. Additionally, the virtual network can be attached to physical devices including network switches, hardware simulations, and intelligent electronic devices (IEDs). Two case studies showcase a relatively complex automatically generated network and an attack on a simple ICS network with an example mitigation.

1. Introduction

Worldwide, societies face increasingly sophisticated malicious cyber-attacks that threaten modern infrastructure and quality of life. The backbone of an industrialized or emerging nation consists of critical infrastructure including energy, water, transportation, and manufacturing. These systems rely on both traditional information technology (IT) and operational technology (OT), the latter having many more security issues due to outdated equipment and poor security controls. This problem is highlighted by the recent ransomware attack on the Colonial Pipeline [1] that forced engineers to shut down the pipeline to prevent the ransomware from spreading to the OT, thus indicating a lack of proper network segregation. To better understand the cyber security issues that plague industrial control systems (ICS) networks, this work presents a set of easy-to-use, inexpensive, flexible configuration tools for modeling and testing ICS networks using the popular network simulator GNS3 [2]. With this tool, different network topologies and networks can be accurately represented and tested for network vulnerabilities, thereby providing a platform for testing mitigation strategies. To support this capability, an easy-to-use configuration file can stand up a large network in seconds and swap configurations just as quickly. Any number of different modules can be used to simulate different devices such as supervisory control and data acquisition (SCADA) systems and intelligent electronic devices (IEDs). These devices are built and deployed as containers, which are a standardized, lightweight method for running applications. More specifically, containers typically require less CPU power, memory usage, and hard drive storage when compared to virtual machines.

1.1 Related Works

A recent survey by Conti et al. [3] discussed a variety of major ICS testbeds developed for research. These testbeds included physical, virtual, and hybrid testbeds. Testbeds in the survey relevant to this work included Minicps [4] and Kypo4industry [5], which were container-based. However, none of these testbeds used GNS3 for representing a network. Also, only one of the testbeds used Docker containers as part of its framework. Khan et al. [6] developed a docker-based testbed for SCADA systems. The testbed by Khan had the advantage of being lightweight in terms of the required computational resources to set up and run the testbed. However, their testbed did not use a high-fidelity model or simulation for the physical system or process under control of the over-all SCADA system. Mlot et al. [7] also used docker for a SCADA tesbed. But their network topology was relatively simple in contrast to the current work, which allows for more complex and automatically generated topologies based on a con-figuration file. Another work uses GNS3 as part of a framework for experimentation with virtual SCADA systems [8]. However, the work does not use a tool to facilitate the creation of complex networks automatically.

Similar to this work, Sandia National Laboratories created a testbed capability involving hybrid testbeds [9] for experiments involving cyber-attacks with Dis-tributed Energy Resources (DERs). The major difference is that Sandia uses virtual machines for emulated devices, whereas this work uses containers. The use of containers may allow for greater portability for quick proof-of-concept experiments on a small computer, such as a laptop.

1.2 ORGANIZATION OF REPORT

This article is organized into the following remaining sections: Section II discusses the motivation for why this tool has been developed. Section III discusses SCADA systems, their use in critical infrastructure, and how they are replicated in this work. Section IV discusses the virtual network used. Section V discusses virtual devices that represent Relays, Meters, and Programmable Logic Controllers (PLCs) among other Intelligent Electronic Devices (IEDs). Section VI describes a case study of the tool for networks involving microgrids and another study involving cyberattacks and physical devices that may be integrated with it. Section VII discusses limitations. Section VIII concludes the article.

2. MOTIVATION

Currently, one of the major issues with evaluating and remediating cyber risks in ICS networks is the sheer cost of the devices typically found in these systems. Device costs can range from thousands to millions of dollars, which is impractical for many research institutions. While some may have access to a system for research use, others have either no access to any system or they have access to a live system from which they are only able to collect data without making any modifications. For those that do have access to a research system, modifications to these systems typically require a lot of time and money due to safety requirements and physical limitations. This makes it difficult to test multiple versions of systems and network structures and limits the overall research capabilities.

For these reasons, several key capabilities are needed for useful vulnerability assessments of emulated ICS networks. First, the emulation platform should be as realistic as possible in regards to the network traffic, switching, and routing. Second, the emulation platform should be simple to set up and allow for testing of different networks with low cost and time requirements to develop test systems. Last, the emulated network should be able to attach to physical devices such as switches, routers, and OT equipment to allow for vulnerability assessment of specific hardware as well as connection to existing physical testbeds. A platform with these capabilities would be very useful for OT security research and were requirements in the creation of this ICS testing platform.

3. SCADA

Supervisory control and data acquisition (SCADA) is a system consisting of hardware and software components that allows different industrial organizations to maintain their operations. These operations can include controlling industrial processes, collecting and processing real-time data, recording events into a log file, and interacting with devices such as pumps, sensors, and motors. SCADA systems help maintain efficiency, process data, and communicate issues to help systems run smoothly.

3.1 MANGO OS FOR SCADA MASTER

One of the goals of this work is to emulate a SCADA system during tests. For this, the SCADA master of choice was Mango OS [10], a SCADA and IIoT platform developed by Radix IoT. Mango can be configured to communicate with a number of different devices using common protocols such as Modbus, DNP3, or ASCII. Mango can poll devices for data simultaneously, while still performing at high speeds. Similar to a PLC, it can monitor devices on the network for events and quickly respond to them, either with automation or manual input, although a PLC is more local to process under control. In traditional SCADA systems, a human-machine interface (HMI) presents the system status to an operator, which is implemented in Mango through the use of dashboards. Mango dashboards provide a graphical interface for visually representing the data and seeing changes in real time. Thus, Mango provides a desirable all-in-one package when it comes to viewing the data, monitoring the devices, and responding to events.

In order to configure the SCADA system, Mango must be configured to request da-ta from the devices on the SCADA network. After that, dashboard visualizations are created to display the measured values. Typically, an HMI displays these values on top of an image that helps represent the SCADA network visually, and Mango allows for a similar representation.

A portion of the dashboard creation is automated using the Mango Configuration Tool, which was developed to ease the process of connecting and configuring the SCADA system. Specifically designed background images can be imported through Mango's user interface (UI), but the Mango Configuration Tool, which is one of tools of this work, has the capability to perform this action automatically with certain dash-board generation options. The capability to write to Modbus registers from the dashboard, as well as view changing values over time, was also added.

3.2 MANGO CONFIGURATION TOOL

The Mango Configuration Tool's primary purpose is to drastically reduce the time it takes to perform these functions manually, make it convenient to spin up new instances across different systems, and allow for human-readable configuration files to be used when designing new environments. The tool is written in Python 3 and utilizes the command-line interface to run commands that interact with Mango through its API. The same functionality that the commands provide can be achieved through Mango's interface in the browser but at a much slower pace. The tool is designed to make the process of loading and viewing data a quicker and more intuitive experience.

The most cumbersome task is loading the data sources and data points into Mango, so it was the first function to be automated. In a real-world environment, the number of devices necessary to add as data sources with their data points would take exponentially longer to complete when compared to using the Mango Configuration Tool. This issue becomes more apparent when similar or identical instances are mimicked across systems or when recreating the same system for multiple tests. To simplify this process, the configuration file lists the data sources and data points to be loaded into Mango. Copying devices is as simple as copying the textual configuration for a device as many times as needed.

The other automation task is visualization of the data. The Mango Configuration Tool can automatically generate a table with each of the data points being listed or a custom dashboard can be created for particular experiments which will then be loaded by the tool. The generated dashboards can then be exported to HTML for usage in other systems or to provide a means for more in-depth manual configuration.

4. VIRTUAL NETWORK

The foundation of the testbed is GNS3 [2]. This platform has been in development for over a decade and allows users to simulate complex networks with much fidelity. Network topologies can include virtualized switches, routers, and machines that run the operating systems of real-world devices. Virtual network connections can be set up identical to physical Ethernet connections, with traffic flowing between devices in much the same way. These virtual connections can be inspected to view actual packets traveling between devices and the data can be saved to PCAP files.

Furthermore, a GNS3 topology can be attached to physical networks, which allows physical hardware to communicate with the virtual network and virtual devices. This capability is crucial for experimentation regarding network threat assessment from a compromised physical device. While emulated devices work well for generalized testing, specific vulnerabilities require these real devices to be present. Integrating these physical devices into the virtual network allows for the best of both worlds in fidelity and flexibility.

However, GNS3 is not a perfect solution. Modeling a small substation with a few power meters, breakers, and a substation controller is easy to do, but larger networks require a good deal of time to set up, even if many of the devices are the same. There is no simple tool for generating larger networks quickly or for resetting a modified network to a known state. If multiple researchers are working on developing the same network on different machines, the large project file must be exported and then imported on the other machines. This also makes version control a challenge.

Due to these issues, the main goal of this work was the development of a tool called GNS3-config which condenses a network topology down to a YAML configuration file and a set of Docker containers described in Section V. Both small and large networks can be expressed entirely in text files and easily managed with version control. Generating a network or resetting it to a known state takes only a few seconds and working with multiple networks is as simple as creating multiple YAML files. With this tool, it is possible to generate various network topologies to simulate different types of ICS networks and SCADA systems. Section VI describes an example topology and its components to show how it can be utilized for cyber resilience tests.

5. VIRTUAL DEVICES

Industrial control networks consist of many different devices that control physical processes. In a power system, these devices monitor power data, control generation, and manage various devices such as circuit breakers. Most of these devices are now able to communicate over Ethernet in addition to legacy serial communications. While it is possible to implement a real network of devices, this is expensive, time-consuming, and difficult to modify. Instead, many of the devices can be emulated to produce realistic data and network communications. For the purpose of vulnerability assessments, flexibility is more useful for testing different configurations and observing a network's resilience.

To emulate devices within GNS3, there are two options: virtual machines or Dock-er containers. A virtual machine (VM) includes an entire operating system (OS) as well as the device emulation software. VMs provide a realistic environment for running software, but it comes with a high computational cost. On the other hand, containers provide an isolated environment for software that utilizes the existing OS's kernel. This inherently limits containers to software capable of running on the GNS3 host's kernel, but in practice this limitation is minimal. For this work, all of the emulation software is developed in Python, which is cross-platform. As such, using Docker containers cuts down on the bloat and increases the total number of emulated devices that can be run at once on a single machine.

Modbus and DNP3 are two of the most common network protocols implemented in SCADA systems [11]. Both protocols use a client and server architecture where the server is a device such as a meter which responds to requests from the client. In this case, Mango is the client for both protocols, so emulated servers for each protocol were implemented in Docker containers. The Modbus device was configured with several coils and holding registers that could be polled by a Modbus client. Likewise, the DNP3 outstation contained several analog inputs from which a master could pull data. This subset of device communication comprised the necessary components for testing network communications in general and observing the effects of various at-tacks on the network.

6. CASE STUDIES

To evaluate the capabilities of the virtual ICS platform, this section presents an example of a relatively complex network topology and another example use case for a virtual power system with physical hardware attached.

6.1 COMPLEX NETWORK TOPOLOGY USE CASE

One of the main purposes of the gns3-config tool is to enable relatively quick proof-of-concept experiments. A contribution of this work is that the tool can automatically generate the required docker containers, their particular characteristics, and the net-work topology that links them. If someone were to implement an actual test bed, the effort expended, the cost of equipment and labor would be far greater. Even other virtualization environments would require some manual and cumbersome configuring.

In this particular use case, it was desired to create a relatively complex topology with multiple switches and nodes to present an OT network for a microgrid. Doing so allows for a demonstration of the breadth of grid devices that could potentially be represented in a virtual environment.

Coordinated Management of Microgrids and Networked and Distributed Energy Resources (COMMANDER) has been a project at Oak Ridge National laboratory with various computerized power system components that include an intellirupter among other devices. It may not be desired to test COMMANDER with cyberattacks, but doing so on a virtual test bed environment is acceptable and allows the user to experiment with different topologies with ease.

Figure 1 shows a topology created using the tool. Figure 2 shows portions of the YAML file that the tool parses to create the topology in GNS3. The topology was loosely inspired by COMMANDER, which had various computerized power system equipment that include photo-voltaic distributed energy resources (DERs), battery storage devices, generators, and intelleruptors. Currently, special devices that provide sensor data through Modbus were used for several of the docker containers. These data represent the behavior of power systems. The GNS network may also be connected to a real-time simulator, such as Typhoon HIL to simulate a given power system.

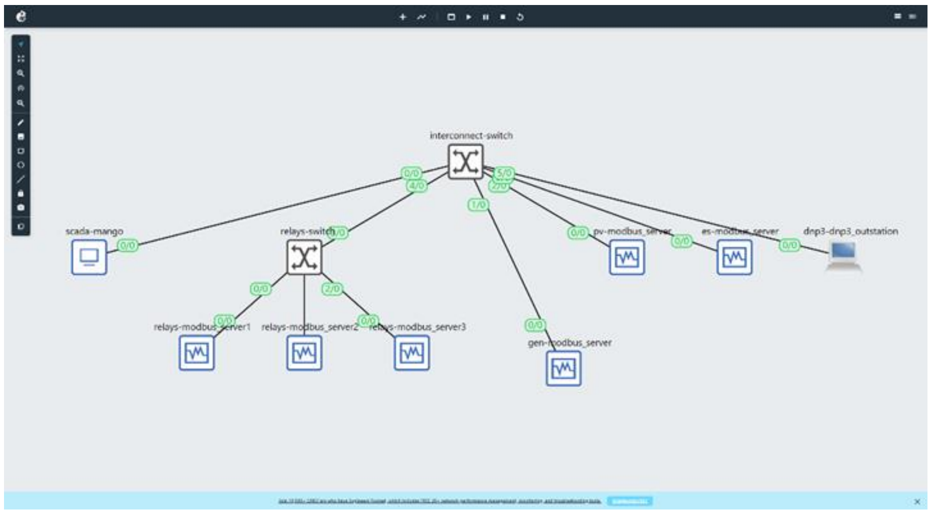
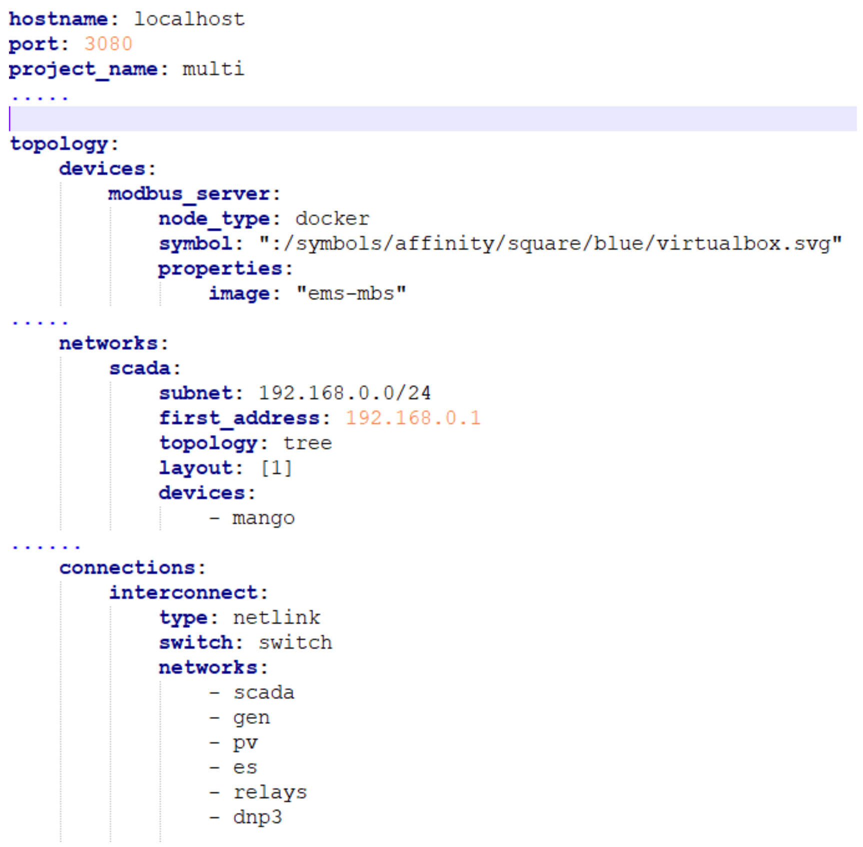


Fig. 1. Diagram showing the virtual devices for a complex network.



```
hostname: localhost
port: 3080
project_name: multi
.....

topology:
    devices:
        modbus_server:
            node_type: docker
            symbol: ":/symbols/affinity/square/blue/virtualbox.svg"
            properties:
                image: "ems-mbs"
.....
    networks:
        scada:
            subnet: 192.168.0.0/24
            first_address: 192.168.0.1
            topology: tree
            layout: [1]
            devices:
                - mango
......
    connections:
        interconnect:
            type: netlink
            switch: switch
            networks:
                - scada
                - gen
                - pv
                - es
                - relays
                - dnp3
```

Fig. 2. Portions of the YAML file to create GNS3 Network

6.2 HYBRID VIRTUAL/ACTUAL HARDWARE USE CASE

For the case in which physical hardware is attached, the network diagram for the system is shown in Figure 3.

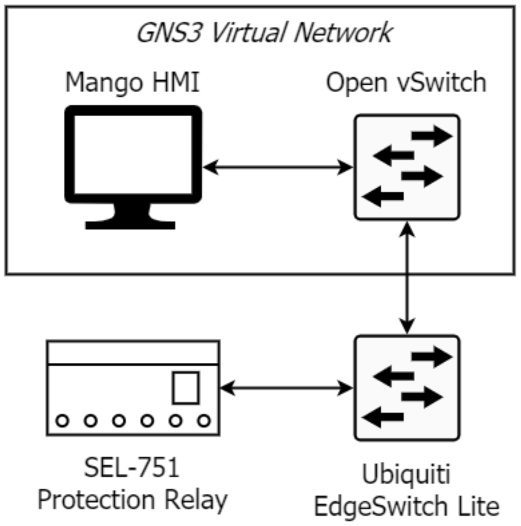


Fig. 3. Diagram showing the virtual devices for a complex network.

The physical hardware used is a protection relay for a power system. The particular model of the relay is an SEL-751 made by Schweitzer Engineering Laboratories (Fig-ure 3). The virtual topology of the network for the power system was defined in a configuration file. Using the file, the tool generated the network in GNS3, which is shown in Figure 5. For this case study, a physical Ubiquiti EdgeSwitch Lite and an SEL-751 feeder protection relay (Figure 4) are connected to the network. For the SCADA system, Mango acts as the HMI and requests specific data from the relay via Modbus. This data is presented to the system operator in a dashboard shown in Figure 6. The dashboard is part of the HMI in the SCADA system that the user sees.

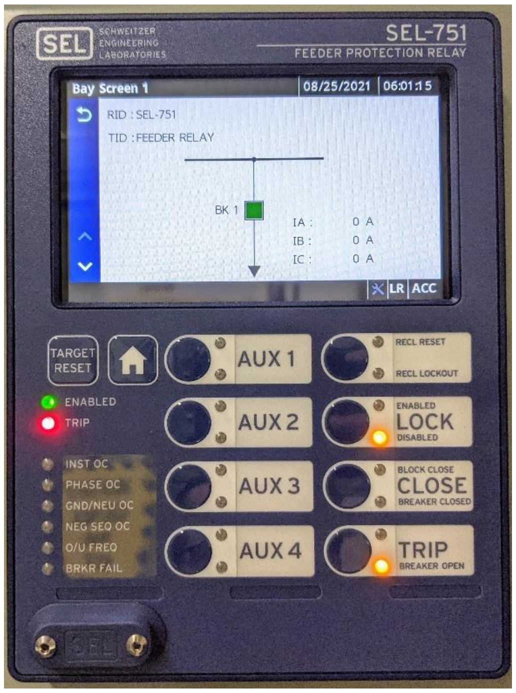


Fig. 4. The SEL-751 Feeder Protection Relay is used to monitor power quality and control a breaker.

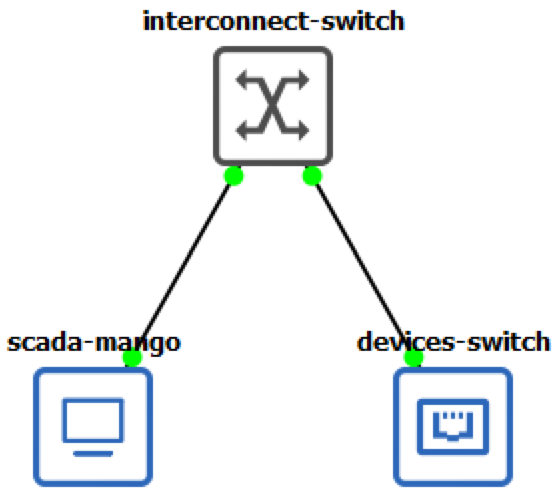
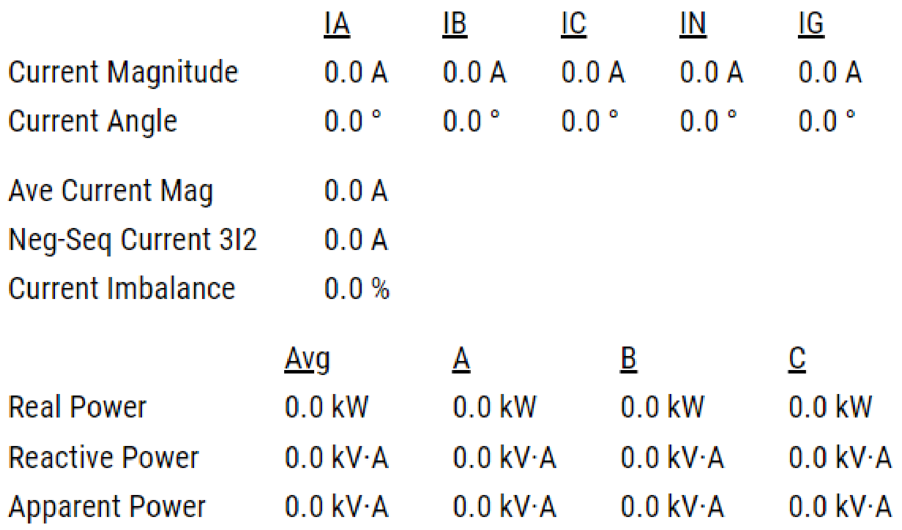


Fig. 5. The SEL-751 Network topology generated in GNS3



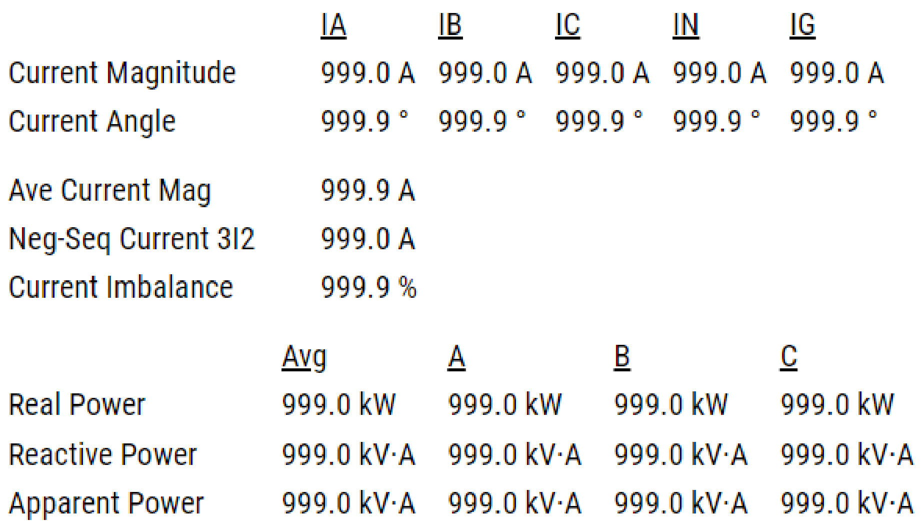
	IA	IB	IC	IN	IG
Current Magnitude	0.0 A	0.0 A	0.0 A	0.0 A	0.0 A
Current Angle	0.0 °	0.0 °	0.0 °	0.0 °	0.0 °

Ave Current Mag	0.0 A
Neg-Seq Current 3I2	0.0 A
Current Imbalance	0.0 %

	Avg	A	B	C
Real Power	0.0 kW	0.0 kW	0.0 kW	0.0 kW
Reactive Power	0.0 kV·A	0.0 kV·A	0.0 kV·A	0.0 kV·A
Apparent Power	0.0 kV·A	0.0 kV·A	0.0 kV·A	0.0 kV·A

Fig. 6. The Mango dashboard shows the current status of the system

As the purpose of the platform is vulnerability assessment, an example network attack was developed. Address Resolution Protocol (ARP) spoofing is a common link layer attack technique that an attacker can use to establish a man-in-the-middle attack. By sending unsolicited ARP responses to two devices, the target devices are led to believe that the attacker is the intended recipient for communications. This attack is performed between Mango and the SEL-751 relay, causing all network traffic to flow During the attack, the dashboard shows false values as in Figure 7. In this example, the values of the Modbus response packets have change to 999.0 before the packets reach the dashboard of HMI. While this is a crafted example, in a real system the attacker can leverage this capability to cause a manipulation of view by modifying the system status to remain constant rather than updating the values. Meanwhile, the at-tacker can cause a manipulation of control by sending a command to the relay to open the breaker, but the dashboard will still appear as if nothing has changed. Without being physically located at the devices, the operator will not realize that the relay has been opened, preventing power from flowing. This effect could have major consequences including loss of power to parts of the system and stress on other power feeders which have to make up for the loss of power. through the attacker's machine, thereby allowing it to be observed and modified.

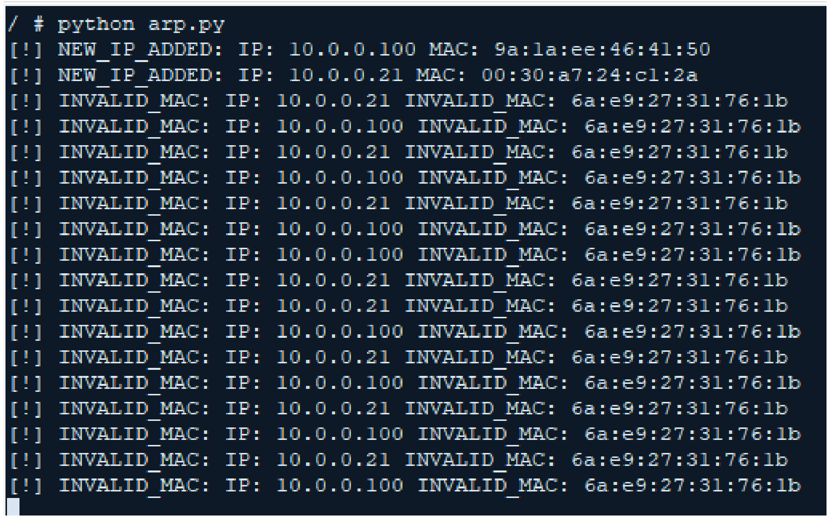


	IA	IB	IC	IN	IG
Current Magnitude	999.0 A	999.0 A	999.0 A	999.0 A	999.0 A
Current Angle	999.9 °	999.9 °	999.9 °	999.9 °	999.9 °
Ave Current Mag	999.9 A				
Neg-Seq Current 3I2	999.0 A				
Current Imbalance	999.9 %				

	Avg	A	B	C
Real Power	999.0 kW	999.0 kW	999.0 kW	999.0 kW
Reactive Power	999.0 kV·A	999.0 kV·A	999.0 kV·A	999.0 kV·A
Apparent Power	999.0 kV·A	999.0 kV·A	999.0 kV·A	999.0 kV·A

Fig. 7. The attacker causes a manipulation of view using an ARP spoofing attack.

To mitigate this attack, there are several options such as static ARP entries, encryption, or active monitoring. In this case, a simple active monitoring system is put in place to observe the ARP traffic and send alerts when new MAC addresses appear on the network. To install this system, a TAP port is created on the virtual switch and connected to the monitoring tool. Now when the attacker attempts to perform ARP spoofing, several alerts appear as shown in Figure 8.



```
/ # python arp.py
[!] NEW_IP_ADDED: IP: 10.0.0.100 MAC: 9a:1a:ee:46:41:50
[!] NEW_IP_ADDED: IP: 10.0.0.21 MAC: 00:30:a7:24:c1:2a
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.21 INVALID_MAC: 6a:e9:27:31:76:1b
[!] INVALID_MAC: IP: 10.0.0.100 INVALID_MAC: 6a:e9:27:31:76:1b
```

Fig. 8. T A monitoring tool alerts when new MAC addresses appear in the ARP traffic.

This example use case showcases the capabilities of the ICS network simulation platform to test network attacks and mitigations. This is done with the GNS3 Config-uration Tool and Mango Configuration Tool, the inclusion of physical hardware com-ponents, and the realistic network simulation. Each of the virtual components is simply a module which can be modified or swapped for various experiments, and new modules can easily be implemented and added. Additionally, physical devices are easily added to and removed from the network via normal switches. Other SCADA systems can be incorporated to better represent a real-world system, and different types of networking switches and routers can be implemented.

7. LIMITATIONS

One of the biggest advantages for physical testbeds is the realism and fidelity necessary to observe certain behaviors when a system is under duress. While this ICS platform does not have the same level of realism, there is a valuable contribution in the realism of the network simulation. While complex physical processes are difficult or impossible to integrate into this system, for many experiments it is not a requirement. For example, a research team may want to test an attacker's ability to pivot across the network from an IED with a known vulnerability and evaluate performance of various network defenses towards mitigation of the network pivoting. In this case, the physical processes are not crucial to the experiment, but the network realism is extremely important. Furthermore, only the vulnerable IED must be physically present, while the rest of the network can be emulated in software so long as they are true to the real-world networking components and defenses.

In this work, no tests were performed to determine how many emulated components could be run simultaneously on a single system. However, based on recent studies such as [12] and [13], it is clear that more containers can be run on a single machine than VMs can be run on the machine. Developing new modules is also faster and simpler within containers as opposed to VMs. Docker is not without its issues though, and these must be considered. First and foremost, running applications within Docker will invariably be slower than running directly on bare metal hardware. This limitation is tempered by containers' ability to isolate multiple processes on the same machine, reducing the overall hardware requirements and thus the cost as well. There are also compatibility issues when running containers as opposed to VMs. Because the native host's kernel is used, the software must be capable of running within the host's kernel. This means Windows-only applications will not run within a Linux kernel and vice versa.

8. CONCLUSIONS

To better secure industrial control systems and critical infrastructure, this paper presents a platform for creating virtual industrial control networks for vulnerability assessment. Built on top of GNS3, an automatic configuration tool can parse a simple, human-readable YAML file to generate large, complex networks in a fraction of the time it would take to set up the physical equivalents. The network consists of containerized modules that represent physical devices. So far, Modbus and DNP3 modules have been created as well as a SCADA module that uses Mango OS. In the future, new modules will be created to allow for testing more systems. To evaluate the security of specific devices and how an attacker might pivot from one device to another via the network, the virtual platform can attach to physical networks and devices. This will allow for a myriad of potential use cases and significantly reduce the time and cost requirements for development and testing of new security measures.

REFERENCES

1. “Darkchronicles: the consequences of the colonial pipeline attack,” Kaspersky, Tech. Rep., May 2021. [Online]. Available: https://ics-cert.kaspersky.com/reports/2021/05/21/darkchronicles-the-consequences-of-the-colonial-pipeline-attack/
2. Gns3. [Online]. Available: https://gns3.com/
3. M. Conti, D. Donadel, and F. Turrin, “A survey on industrial control system testbeds and datasets for security research,” arXiv preprint arXiv:2102.05631, 2021.
4. D. Antonioli and N. O. Tippenhauer, “Minicps: A toolkit for security research on cps networks,” in Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy, 2015, pp. 91–100.
5. P. Celeda, J. Vykopal, V. Svabensky, and K. Slavıcek, “Kypo4industry: A testbed for teaching cybersecurity of industrial control systems,” in Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 2020, pp. 1026–1032.
6. M. Khan, O. Rehman, I. M. Rahman, and S. Ali, “Lightweight testbed for cybersecurity experiments in scada-based systems,” in 2020 International Conference on Computing and Information Technology (ICCIT-1441). IEEE, 2020, pp. 1–5.

7. E. D. G. Mlot, J. Saldana, and R. J. Rodr´ıguez, “Towards a testbed for critical industrial systems: Sunspec protocol on der systems as a case study,” in 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE, 2022, pp. 1–4.
8. A. Dehlaghi-Ghadim, A. Balador, M. H. Moghadam, H. Hansson, and M. Conti, “Icssim—a framework for building industrial control systems security testbeds,” Computers in Industry, vol. 148, p. 103906, 2023.
9. V. Urias, B. Van Leeuwen, and B. Richardson, “Supervisory command and data acquisition (scada) system cyber security analysis using a live, virtual, and constructive (lvc) testbed,” in MILCOM 2012-2012 IEEE Military Communications Conference. IEEE, 2012, pp. 1–8.
10. Mango os. [Online]. Available: https://mango-os.com/
11. W. Xu, Y. Tao, and X. Guan, “The landscape of industrial control systems (ICS) devices on the internet,” in Conference on cyber situational awareness, data analytics and assessment, 2018.
12. A. M. Potdar, D. Narayan, S. Kengond, and M. M. Mulla, “Performance evaluation of docker container and virtual machine,” Procedia Computer Science, vol. 171, pp. 1419–1428, 2020.
13. K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, “Performance comparison analysis of linux container and virtual machine for building cloud,” Advanced Science and Technology Letters, vol. 66, no. 105-111, p. 2, 2014.