VICSORT - A Virtualised ICS Open-source Research Testbed

1st Conrad Ekisa, 2nd Diarmuid Ó Briain and 3rd Yvonne Kavanagh
engCORE Research Centre,
Department of Aerospace, Mechanical & Electronic Engineering,
Institute of Technology, Carlow, Ireland
Email: conrad.ekisa@itcarlow.ie

2022 Cyber Research Conference - Ireland (Cyber-RCI) | 978-1-6654-7422-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/Cyber-RCI55324.2022.10032670

***Abstract*—Industrial Control Systems (ICS) are at the forefront of most, if not all the critical infrastructure and critical service delivery. ICS underpin modern manufacturing and utility processes and greatly contribute to our day-to-day livelihoods. However, there has been a significant increase in the number and complexity of cyberthreats specifically targetted at ICS, facilitated by increased connectivity in an effort to improve production efficiency. Furthermore, the barriers of entry to ICS cybersecurity are still high given the limited skills base, expensive and proprietary hardware and software as well as the inherent dangers of manipulating real physical processes. This greatly inhibits the practical application of cybersecurity tools in ICS environments and therefore the opportunity for practitioners to gain valuable ICS cybersecurity experience. ICS Testbeds are often either expensive and are not necessarily holistic enough to provide learners with the complete breadth of ICS. This paper introduces VICSORT, a open-source virtualised ICS testbed that provides a platform for ICS cybersecurity learners and practitioners to interface with an ICS environment that closely emulates a real-world ICS, as well as explore and practice techniques for attack and consequently defence of an ICS. VICSORT builds upon the Graphical Realism Framework for Industrial Control Systems (GRFICS) to offer an easier to deploy environment with greater flexibility, whilst requiring significantly less resources all reducing the cost to the learner.**

***Index Terms*—ICS, Cybersecurity, Open-source, Virtualised, Testbed**

I. Introduction

ICS, in some form, have existed since the 20th century and certainly long before the advent of the Internet and connectivity were as prevalent as they are today. At the time, the most common ICS security design principle was *security by obscurity*. This means that if a system is air-gaped, that is, contained with no outside connections, then the system is inherently secure. However, over the years, in an effort to provide more visibility, efficiency and information about the industrial processes, there is more connectivity both to, and within, the ICS zone as well as greater synergy between Enterprise and Control zones. Advantageous as this can be, new ICS cyber attack vectors are exposed. This, coupled with the fact that ICS were initially designed with a *security by obscurity* principle, has left many ICS vulnerable to attack by malicious actors. Due to the critical nature of a number of ICS, there is now a understanding of the need to drive towards building and growing ICS cybersecurity aptitude and competence to better protect these systems.

Funded by the 2021 Government of Ireland - International Education Scholarship

However, the barriers of entry to ICS cybersecurity are still high given the limited skills base, expensive and often proprietary hardware and software, as well as the inherent dangers of manipulating real physical processes. These factors serve to greatly inhibit new entrants to the ICS cybersecurity space and became the motivation underpinning this research work. Conti et al [1] describe three types of testbeds: physical, virtual and hybrid testbeds. The focus of this paper shall be on the VICSORT virtual testbed.

This paper wishes to acknowledge and recognise the work of the Georgia Institute of Technology and Fortiphyd Logic [2] in the development of the Graphical Realism Framework for Industrial Control Systems (GRFICS) testbed which provided a foundation upon which VICSORT is built. The GRFICS testbed made the following contributions to the ICS cybersecurity research space:

- Conversion of the simplified Tennessee Eastman (TE) Challenge process simulation into a more portable and accessible format.
- Novel 3D visualisation of a dynamic chemical process simulation to increase engagement and realism.
- The most complex and complete virtualisation of an ICS network to date, released Free, Libre and Open-Source Software (F/LOSS)
- Modular framework for easy expansion or conversion to other physical processes and protocols.

This paper presents VICSORT, a virtualised ICS open-source research testbed that builds upon the GRFICS testbed with the following contributions:

- An all-in-one F/LOSS based testbed rebuilt with GNU/LinuX Containers (LXC) to provide a leaner overall build requiring significantly less system resources to operate. The testbed is easily deployable offline or online on the Cloud.
- A detailed guide on how to reproduce the testbed from scratch to provide a more in-depth understanding to how the ICS components operate.
- A number of modifications to the testbed components including Python library upgrades/replacements to the currently outdated and publicly unavailable libraries used

in the GRFICS version, Operating System (OS) upgrades, firewall upgrades to more closely mimic a real-world ICS, as well as architectural and design modifications.

The remainder of this paper is organised as follows: Section II provides a background on ICS giving an overview of the general architecture, ICS components as well as the common protocols used to link systems. Section III discusses related ICS testbed work as well as challenges with the GRFICS testbed that the VICSORT testbed addresses. Section IV discusses the VICSORT testbed describing its network topology, components involved as well as enhancements included in this testbed. Section V provides an evaluation of the testbed. Section VI highlights the contributions made by VICSORT as well as referencing where and how the testbed can be accessed. Section VII finally provides a conclusion to this paper.

II. Background

A. Industrial Control Systems

This section briefly introduces the main components of an ICS from which VICSORT is derived. In previous work [3], ICS is introduced as a broad umbrella term that encompasses Supervisory Control And Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control devices such as Programmable Logic Controllers (PLC). These are interconnected either via serial connections or, as is becoming more popular, deterministic Ethernet networks, using the Transmission Control Protocol (TCP) / Internet Protocol (IP) (TCP/IP) protocols, to acquire data from sensors and control mechanical actuator devices. These systems are usually quite complex and include heterogeneous hardware and software components and processes being controlled or monitored, computational nodes, communication protocols, SCADA systems and controllers.

a) ICS Architecture: The ICS architecture is generally defined by the Purdue Model [4] as depicted in Fig 1. A discussion on each of these levels is highlighted in previous work [3]. This paper uses this model as an architectural model from which VICSORT is built.

b) ICS Components: ICS are composed of a wide range of devices that serve specific roles. Specifically, this paper will reintroduce five primary ICS components, previously highlighted in [3], that are implemented in the VICSORT design. These are the PLC, Human Machine Interface (HMI), engineering workstation, and ICS field devices.

c) ICS Protocols: A number of protocols exist within the ICS space; however, this paper will focus on Modbus/TCP which is implemented in the VICSORT implementation.

Modbus is a serial communication protocol initially published by Modicon (now Schneider Electric) in 1979 for use with its PLCs. It was later adapted for use over TCP/IP. Modbus is an application layer messaging protocol, positioned at Layer 7 of the Open Systems Interconnection (OSI) model, that provides client/server communication between devices connected on different types of buses or networks [5].

VICSORT leverages Modbus/TCP on port 502 for communication between the field devices, PLC and HMI.

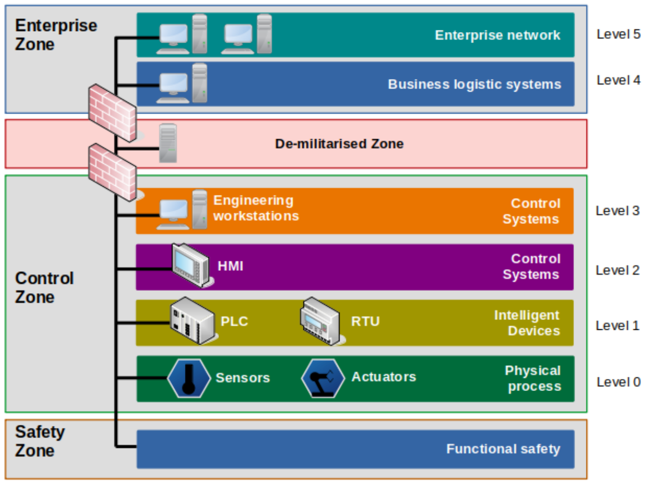


Fig. 1. ICS Purdue Model

III. Related Work

A. Previous Testbeds

In previous work [3], six ICS Testbed models were highlighted from Thiago et al [6], Genge et al [7], Maynard et al [8], Giani et al [9], David et al [2] and Bertrand et al [10].

Furthermore, other testbeds such as Hui et al [11], Candel et al [12], Korkmaz et al [13] and Gardiner et al [14] have also been reviewed.

All these testbeds exhibit great individual strengths such as closely modelling ICS operations, integrating major ICS components, as well as virtualising ICS components for easier deployment and modification. However, similar constraints surround a number of them such as non-replicability due to non-publicly available source code, lack of a realistic visual element to the ICS physical process, requirement to purchase specific physical components to implement the ICS physical process, or lack of a holistic representation of an ICS from the testbed models.

B. Testbed Technical Requirements

In an effort to produce a testbed model that is reproducible, easily deployable, scalable, lean in terms of compute resources, as well as offering a contribution towards practical hands-on ICS cybersecurity training, the following testbed technical requirements were set out [3]:

- The testbed integrates major OT components such as PLCs, HMIs, engineering workstations, firewalls and a physical process.
- The testbed incorporates the Modbus/TCP ICS protocol given their prevalence in real-world ICS implementations today.
- The testbed architectural build closely follows the well known ICS Purdue model.
- The testbed is built using F/LOSS tools to facilitate ease of implementation, and offers potential for future enhancements.

- The testbed incorporates a 3D visualisation to simulate the ICS control process. This is to allow for visualisation of the physical consequences of a successful cyber attack on an ICS.
- The entire testbed build is compute resource lean. The resource constraints set for this project were 16GB RAM, 100GB HDD, 4CPUs @ 2.4GHz, 4GB dedicated graphics card.

C. GRFICS

As cited in previous work [3], David et al [2] propose the Graphical Realism Framework for Industrial Control Systems (GRFICS). Developed by researchers from Georgia Institute of Technology and Fortiphyd Logic, GRFICS [15] is a open source ICS simulation tool based on the TE process [16] with a goal of bringing practical ICS security skills to a wider audience. The testbed, built using Python on GNU/Linux, is currently designed for educational purposes and offers only a single ICS process, that is to say, the TE process. Fortiphyd Logic have built similar simulations, available commercially, on their training portal. The GRFICS platform is comprised of a total of five Virtual Machines (VMs), built on the Oracle VirtualBox hypervisor that perform the functions of; a PLC, a HMI, a firewall, an engineering workstation and a novel 3D visualisation of the physical process [15]. The testbed also includes a network setup to be implemented within the Oracle VirtualBox hypervisor. The PLC is implemented using *OpenPLC* [6], the HMI is implemented using *ScabaBR* [17], the firewall is implemented with *pfSense* [18], an engineering workstation that is a standard workstation with a GNU/Linux Operating System (OS) and the software to make changes to the PLC, and lastly, the 3D physical process simulation is implemented with Unity Game Engine [19]. The testbed supports the running of a number of ICS related attacks such as Man in the Middle (MitM) attacks, Command Injections, False Data Injection, PLC Reprogramming, Loading Malicious Binary Payloads, and common IT attacks such as password cracking. While the GRFICS testbed is prebuilt, the source-code is available and can be customised or modified to meet the user's requirements.

D. VICSORT contributions to GRFICS

VICSORT improves the fundamental GRFICS build to provide for a smaller, leaner and a more easily deployed testbed build. Specifically, VICSORT furthers the work achieved in GRFICS by providing:

- An all-in-one open-source testbed rebuilt with LXD container manager and Kernel-based Virtual Machine (KVM) hypervisor to provide a leaner overall build in terms of required compute resources, that's easily deployable locally or on a Cloud deployment.
- A publicly available how-to guide detailing how to reproduce the testbed to provide a more in-depth understanding to the testbed build process as well as how ICS components operate and work together.
- A number of modifications to the testbed components including Python library upgrades/replacements to the currently outdated and publicly unavailable libraries used in the previous version, OS upgrades, firewall upgrades to more closely mimic a real-world ICS, as well as architectural and design modifications.

IV. VICSORT

VICSORT is a modified build of the GRFICS testbed that seeks to lower the barrier of entry to ICS cybersecurity even further as well as contribute to practical hands-on ICS cybersecurity training. VICSORT attempts to provide a holistic overview of an ICS environment integrating the major components of an ICS namely; the PLC, HMI, engineering workstation, firewall and physical process. VICSORT virtualises all these components and builds a network topology to facilitate any required communication between the participating nodes. The previous section introduced the contributions made by VICSORT and this section shall expound on these contributions in more detail. VICSORT, built on GNU/Linux OS, heavily leverages LXC with an LXD container manager, Python programming, and KVM for its implementation. While a fair understanding of these elements necessary to fully appreciate the VICSORT build process, this is not absolutely required to run the testbed. This testbed is designed to support ICS cybersecurity students, new entrants to the ICS cybersecurity space and industry practitioners looking to reskill or upskill in ICS cybersecurity.

This section shall discuss various aspects of the testbed including the testbed architecture, testbed components and initialisation procedures.

A. Testbed Architecture

The testbed attempts to closely mimic the Perdue model whilst highlighting the major components of an ICS. As depicted in Fig 1, the Purdue module divides a manufacturing company or utility network into six levels broadly categorised under the Enterprise, De-militarised and Control Zone. These levels were described in previous work [3].

This testbed focuses on the Control Zone, as depicted in Fig 2. This implementation highlights four levels in the Purdue model:

- *Level 0*: This includes sensors and actuators, usually termed as field devices, that interact with the actual physical processes, also found at this level.
- *Level 1*: This is the basic control level and is made up of PLCs, Remote Terminal Units (RTU) that control the physical devices.
- *Level 2*: This is the area supervisory control level and is typically comprised of HMIs at the shop or factory floor. End-user or service engineers can interface with them. Alarm systems and control room workstations are also found at this level.
- *Level 3*: This is the site operations level that typically houses systems that supports plant control operations

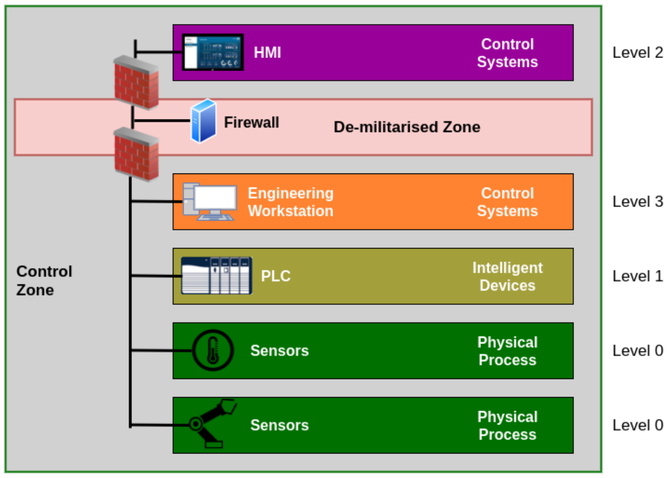


Fig. 2. VICSORT Purdue Model Implementation

like a file sharing server, data historian, reporting and scheduling systems.

VICSORT v1.0 will focus on the Control Zone and may grow to incorporate the Enterprise zone in future iterations.

B. Testbed Network Topology

From this point forward, the baremetal computer or VM hosting VICSORT, will be termed as 'host-compute'.

VICSORT is divided into two networks namely the ICS or Control Zone network and the De-militarised Zone (DMZ) network. As illustrated in 3, the VICSORT network topology, the Control Zone is assigned to the 192.168.95.0/24 IP subnet and the DMZ is assigned to the 192.168.90.0/24 IP subnet. During normal testbed operation, communication between the two subnets and more specifically, communication between nodes is managed via the firewall. All traffic within the testbed is routed through the firewall. The testbed comprises of five nodes with a static IP address mapping as listed in Table I. The testbed also includes an attacker node that has successfully breached the DMZ Zone. This attacker node obtains an IP address via Dynamic Host Control Protocol (DHCP) on the DMZ network. However, the DHCP Server is not included in the setup but operates in the background.

TABLE I
VICSORT IP ADDRESS MAPPING

Node	IP Address Mapping
HMI	192.168.90.5 /24
Firewall	- WAN: 192.168.90.100 /24 - LAN: 192.168.95.100/24
PLC	192.168.95.2 /24
Engineering Workstation	192.168.95.5 /24
Plant Simulation	192.168.95.10 - 15 /24
Attacker	192.168.90.XX /24

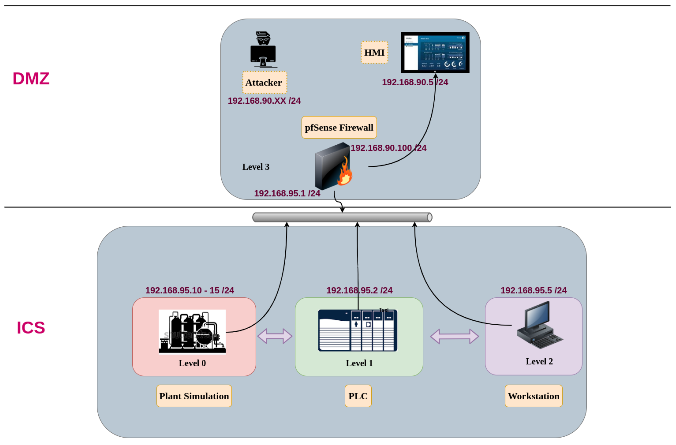


Fig. 3. VICSORT Network Topology

C. Testbed Components

As already outlined, the testbed comprises of a total of six components. This section will describe each of these components:

a) HMI: The HMI function is hosted in a LXC with an independent Ubuntu Linux 20.04 LTS OS. The HMI is implemented in software using *ScadaBR* [17] and resides within the DMZ – 192.168.90.0/24 network. During normal operation, the HMI is accessible from the host-compute via *http://192.168.90.5:9090/ScadaBR*. The HMI is referred to as **hmi-container** within VICSORT.

b) PLC: The PLC function is also hosted in a LXC with an independent Ubuntu Linux 20.04 LTS OS. The PLC is implemented in software using *OpenPLC* v2 [6] and resides within the Control Zone – 192.168.95.0/24 network. During normal operation, the PLC is accessible from the host-compute via *http://192.168.95.2:8080*. The PLC is referred to as **plc-container** within VICSORT.

c) Engineering Workstation: Additionally, the Engineering Workstation function is hosted in a LXC with an independent Ubuntu Linux 20.04 LTS OS. This node is specifically intended for the creation of PLC programs using ladder logic edited using *OpenPLCEditor* [20] and the completed programs can then be uploaded to the PLC. For this reason, this node is hosted on the same network as the PLC. This node resides within the Control Zone – 192.168.95.0/24 network. The Engineering Workstation node features a Graphical User Interface (GUI) to allow a user to interface with the node and create or modify PLC logic. The Engineering Workstation node is referred to as **workstation-container** within VICSORT.

d) Physical Process: The physical process is hosted in a LXC with an independent Ubuntu Linux 16.04 LTS OS. This node combines a physical process simulation that runs on the Unity Gaming Engine platform and virtualised sensors communicating using Modbus. Specifically, the simulation is implemented with Unity's WebGL that facilitates the simulation to be run via an Apache Tomcat web server hosted locally on the container.

TABLE II
PHYSICAL PROCESS IP ADDRESS MAPPING

IP Address	Port Bindings	Program Name	Function
192.168.95.10/24	502	Python	feed1.py
192.168.95.11/24	502	Python	feed2.py
192.168.95.12/24	502	Python	purge.py
192.168.95.13/24	502	Python	product.py
192.168.95.14/25	502	Python	tank.py
192.168.95.15/24	502	Python	analyzer.py
0.0.0.0	55555	Simulation	simulation server that updates 3D visualisation
0.0.0.0	80	Apache2	access the simulation via web browser

The simulation is designed around the TE Challenge Process [16] and simulates a reactor core, input elements *A* and *B*, as well as a Purge output and the Product output. The TE Challenge Process is further discussed in the subsection IV-E.

Consequently, six virtualised sensors that generate readings displayed within the simulation are also hosted on this node. These sensors, all implemented using the *pymodbus* Python library are *feed1.py*, *feed2.py*, *product.py*, *purge.py*, *analyzer.py* and *tank.py* with an IP address mapping as listed in Table II.

This node resides within the ICS Zone – 192.168.95.0/24 network. The simulation is accessible via *http://192.168.95.10*. This node is referred to as **simulation-container** within VICSORT.

e) Firewall: The firewall of choice for this testbed is *pfSense* [21] that runs on FreeBSD [22] OS. Within VICSORT, *pfSense* is hosted on a VM and is attached to the ICS and DMZ networks. The IP address mapping within the ICS zone is 192.168.90.100/24 and 192.168.95.100/24 within the DMZ. The firewall has custom firewall rules present upon deployment via a *base_firewall_configs* file. The firewall implementation associates the ICS zone with the Local Area Network (LAN) and the DMZ with the Wide Area Network (WAN). The WAN is intended to face the Enterprise network whilst the LAN faces the Control zone.

The following rules on the WAN within the ruleset:

1) Allow all communication from the HMI to PLC
2) Allow access to the *OpenPLC* Web User Interface (UI) from the WAN network
3) Allow access from WAN to simulation VM web interface
4) Allow Internet access for all nodes on the WAN. This rule is disabled by default and should be enabled when Internet access is required on the WAN nodes for example for repository updates or further package downloads. However, the attacker node is exempt from this rule and always has Internet access.

The following rules, associated with the LAN are included within the ruleset:

1) Allow Internet Control Message Protocol (ICMP) to firewall from the 192.168.95.0/24 (LAN) network. This allows nodes on the LAN to ping the firewall
2) Allow all communication from the PLC to HMI
3) Allow Internet access to all nodes on the LAN. This rule is disabled by default and should be enabled when Internet access is required on the WAN nodes for example for repository updates or further package downloads.

All the nodes, except the attacker node, have their default route pointing to the firewall. Therefore, traffic from these nodes is all routed via the firewall. The *pfSense* web UI is accessible via *http://192.168.95.100*. This node is referred to as **pfsense-vm** within VICSORT.

f) Attacker: The attacker node integrated into VICSORT is Kali Linux 2021 running in an LXC. This node contains the entire Kali Linux suite of tools as well as a GUI accessible via Remote Desktop Protocol (RDP). This node resides in the DMZ and obtains an IP address dynamically from a DHCP server reading within that network. The assumption is that the attacker has successfully breached the enterprise network and managed to pivot to the DMZ zone. This node is referred to as **attacker-container** within VICSORT.

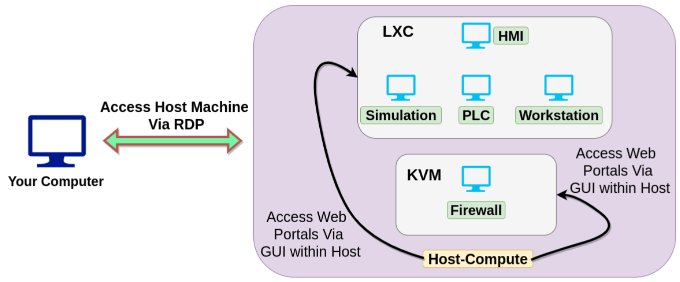
D. Testbed Underlying Build Design

The host-compute housing VICSORT is built based on the following criteria:

a) Internal Design: VICSORT relies heavily on LXC/LXD and KVM for its implementation. There are a total of five LXC and one VM in this implementation. The foundational technologies used in the development of VICSORT are Python and GNU/Linux. The networking between all these nodes is handled by the LXD networking function that creates two network bridges within the host-compute in accordance to the VICSORT network topology namely *icszone* and *dmzzone*. To maintain a low OS resource utilisation, VICSORT employs *lubuntu-desktop* [23] that is a fast and light-weight GUI for Ubuntu. The testbed also contains initial setup files available within the project Github repo that are necessary during the testbed's setup.

b) Intended Usage: This testbed is designed to be self-contained. This means that any Universal Resource Locaters (URL) used within the tested, for example the HMI URL, or URL to the simulation are intended to be accessed within the host machine. This testbed is designed to be suitable for deployment within a Type 2 hypervisor such as VirtualBox or VMWare, or on any cloud infrastructure. It should be noted however that these URLs and testbed resources can be accessed outside the host machine though this is not covered within the scope of this project. It is recommended to use one of these two options, either a Type 2 hypervisor or Cloud to host VICSORT as opposed to hosting the testbed directly on a Personal Computer (PC).

The host-compute contains a GUI to facilitate access to the various components of the testbed, many of which are accessible via a web browser. If the host-compute is within the Cloud, a user is required to RDP from their computer to the host-compute. Within this session, the user will be able to use the web browser to access web portal utilities that most components leverage for their operation. If the host-compute is within a Type 2 hypervisor, a user can simply utilise the console interface provided by the Type 2 hypervisor itself to access the host-compute, as opposed to using RDP. However, RDP is still an option in this case. Furthermore, the attacker container will also be accessed via RDP as its envisaged that the attacks will be initiated from the attacker node and not the host-compute. Figure 4 is a schematic illustrating how access to VICSORT, and the nodes within it, is managed.



Access Host Machine Via RDP
Your Computer
LXC
HMI
Simulation
PLC
Workstation
KVM
Firewall
Access Web Portals Via GUI within Host
Access Web Portals Via GUI within Host
Host-Compute

Fig. 4. Accessing the GUI components of VICSORT

c) Testbed Initialisation: The testbed contains a number of nodes. This warranted the need for an initalisation script to bring the testbed to an operational state upon boot of the host-compute. Upon boot of the host-compute, LXD is in a running state as a *systemd* service. The LXCs in most cases will all be in a running state too. The initialisation script initialises the testbed performing the following functions in chronological order.

The script obtains IP addresses for all the LXC. These IP addresses are static; however, they still need to be requested from the respective DHCP server. The script will then output all the containers and their corresponding IP addresses, start the firewall VM within KVM, provision the HMI and start *ScadaBR*, provision the PLC and start *OpenPLC*, provision the simulation node, start the simulation and pymodbus sensors, set the correct timezone within all the nodes, and finally configure a new default route on all the nodes except the attacker, pointing all their traffic to the firewall.

At this stage, a user is ready to start using the testbed.

E. Tennessee Eastman Challenge Process

The TE Challenge Process is based off the TE plant. Figure 5 illustrates the TE plant that the presents the challenge to establish control over.

The TE plant wide Industrial Control Process problem was proposed by Downs and Vogel in 1993 [24] as a challenge test problem to several control related topics such as multivariable controller design, optimisation, adaptive and predictive control and non-linear control. Since its design, over 60 studies have used this case study for alternative plant-wide control, process monitoring, fault detection and identification. The TE process is a realistic simulation environment of a real chemical process that is comprised of an exothermic two-phase (liquid and vapour) reactor, a flash separator, recycle compressor, and a reboiled stripper.

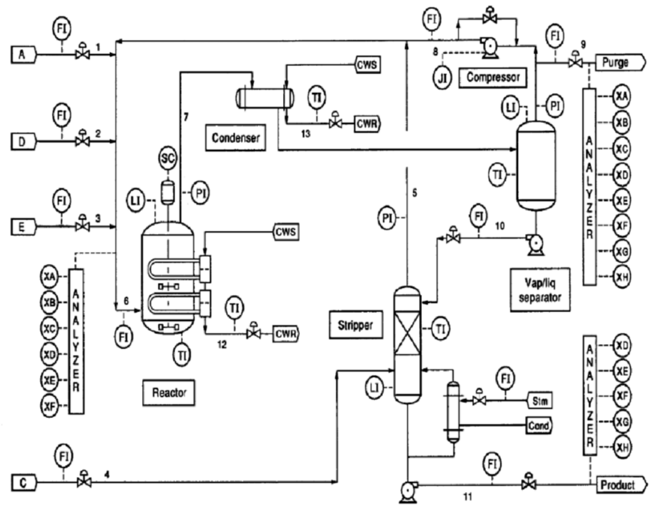


Fig. 5. Tennessee Eastman Challenge Process [24]

From Figure 5, the gas reactants A, C, D and E enter the reactor where the reactants undergo an irreversible exothermic (which means the temperature gradually increases) catalytic gas phase reaction. Since the reaction is exothermic in nature, it is cooled by the Cold Water Supply (CWS) and Cold Water Reset. Inert gas does not undergo chemical reactions under a set of given conditions. The partial condenser recovers the products from the reactor exit gas stream. The stripper is used to minimise the loss of reactants D and E in the liquid product stream. The gas overhead from the stripper is combined with the compressed overhead from the separator and recycled back to the reactor. The purge stream is used to prevent build-up of excess reactants, the inert B and the by-product F. The process produces two products from four reactants. Also present are an inert and a by-product making eight components A, B, C, D, E, F, G and H. The process variables are temperature, pressure, level and flow rate.

VISCORT simplifies these process with only two inputs; *feed1* and *feed2* as well as a Product output and a Purge byproduct. The process variables measured are pressure (kPa) and Level (%) and flowrate (kMol/h). Figure 6 illustrates the physical process simulation as well as its representation as shown on the HMI.

F. Testbed in Operation

Figure 7 lists the five LXC and single KVM VM that make up VICSORT, in an operational state. Figure 6 illustrates the physical process simulation as well as its representation as shown on the HMI.

V. BENCHMARKS

This section documents system benchmarks achieved by the testbed.

A. GRFICSv2 and VICSORT – System requirements

As previously mentioned, this testbed builds on the work achieved in the GRFICSv2 [25] project that utilised VirtualBox

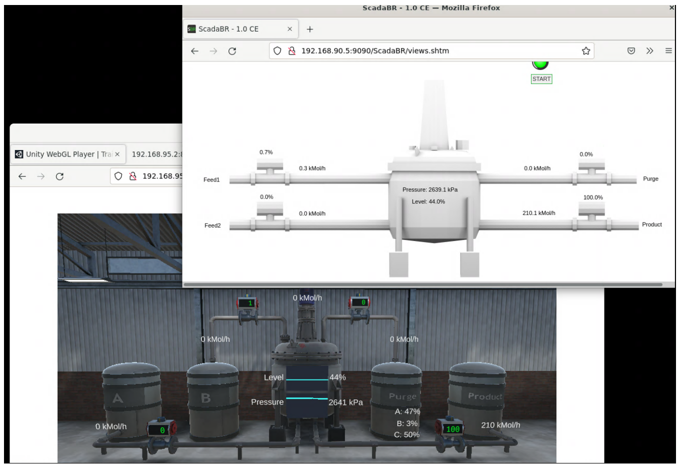
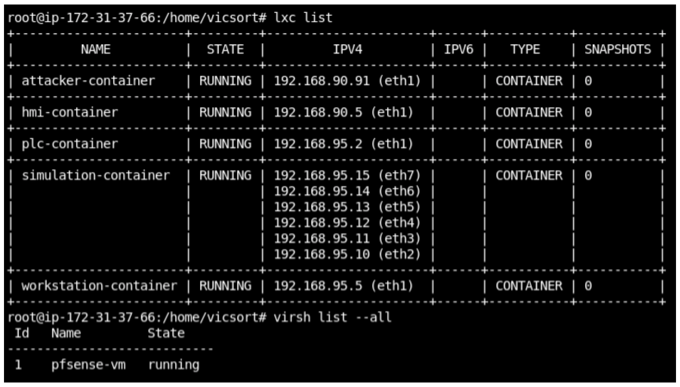


Fig. 6. HMI and Physical Process Simulation



```
root@ip-172-31-37-66:/home/vicsort# lxc list
+---------------------+---------+---------------------+------+-----------+-----------+
|        NAME         |  STATE  |        IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+---------------------+---------+---------------------+------+-----------+-----------+
| attacker-container  | RUNNING | 192.168.90.91 (eth1) |      | CONTAINER | 0         |
+---------------------+---------+---------------------+------+-----------+-----------+
| hmi-container       | RUNNING | 192.168.90.5 (eth1) |      | CONTAINER | 0         |
+---------------------+---------+---------------------+------+-----------+-----------+
| plc-container       | RUNNING | 192.168.95.2 (eth1) |      | CONTAINER | 0         |
+---------------------+---------+---------------------+------+-----------+-----------+
| simulation-container | RUNNING | 192.168.95.15 (eth7) |      | CONTAINER | 0         |
|                     |         | 192.168.95.14 (eth6) |      |           |           |
|                     |         | 192.168.95.13 (eth5) |      |           |           |
|                     |         | 192.168.95.12 (eth4) |      |           |           |
|                     |         | 192.168.95.11 (eth3) |      |           |           |
|                     |         | 192.168.95.10 (eth2) |      |           |           |
+---------------------+---------+---------------------+------+-----------+-----------+
| workstation-container | RUNNING | 192.168.95.5 (eth1) |      | CONTAINER | 0         |
+---------------------+---------+---------------------+------+-----------+-----------+
root@ip-172-31-37-66:/home/vicsort# virsh list --all
 Id   Name         State
----------------------------
 1    pfsense-vm   running
```

Fig. 7. VICSORT Containers and VM

VMs for its testbed implementation. Table III illustrates a breakdown of the system resources required in the GRFICSv2 implementation. Notable is that at-least 5GB RAM is required on the host-compute to optimally run the testbed. This utilisation increases when web browsers are launched to access the various web UIs in the testbed. GRFICS also relies on VirtualBox's networking function to manage communication between the VMs and host-compute. Figure 8 illustrates a VirtualBox screenshot with all six GRFICSv2 VMs running.

VICSORT makes OS upgrades to the GRFICS implementations with the PLC, HMI and engineering workstation containers running Ubuntu Linux 20.04 LTS, the *pfSense* VM running FreeBSD 12.2 and the attacker container running Kali Linux 2021.2

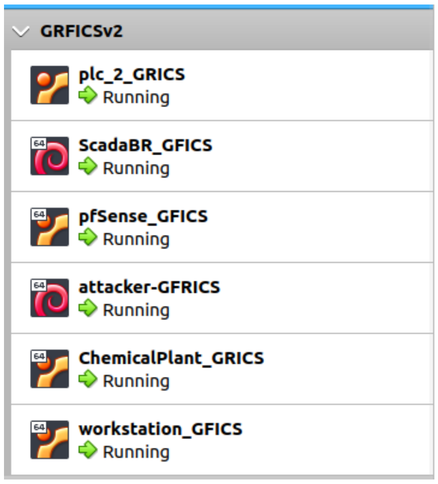
Figure 9 lists the total hard disk space occupied by VICSORT as 23GB – a significant reduction from the 38.69GB occupied by GRFICSv2.

B. VICSORT RAM Utilisation

Significant contributions were made regarding the RAM utilisation required for the testbed to run. Figure 10 puts the RAM of host-compute just after boot at about 1GB. It should be noted that this deployment was on an Amazon Web Services (AWS) instance. Figure 11 then illustrates the RAM utilisation with the testbed running at 2.31GB resulting in a total testbed RAM utilisation of about 1.2GB - a notable reduction from the 5GB required by GRFICSv2.

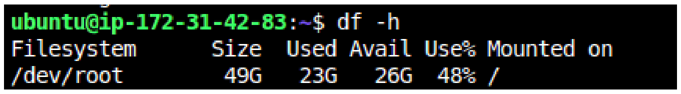
TABLE III
GRFICSv2 System Resources

Virtual Machine	RAM (MBS)	Virtual HDD Size (GB)	Actual HDD Space Used (GB)	Operating System
PLC VM	512	8	3.28	Ubuntu 16.04
HMI VM	1024	100	2.27	Debian 9
PfSense Firewall	512	5	1.14	FreeBSD 11.3
Simulation VM	1024	6.5	3.93	Ubuntu 16.04
Engineering Workstation	1024	10	7.34	Ubuntu 16.04
Attacker PC	1024	80	20.73	Kali Linux 2020.4
Total	**5 GB**	**209.5**	**38.69**	



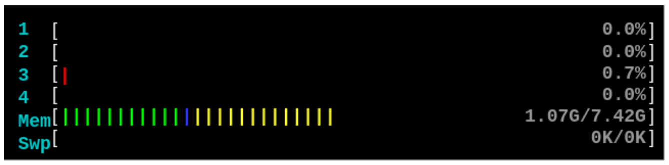
GRFICSv2
plc_2_GRICS
Running
ScadaBR_GFICS
Running
pfSense_GFICS
Running
attacker-GFRICS
Running
ChemicalPlant_GRICS
Running
workstation_GFICS
Running

Fig. 8. GRFICSv2 Testbed running



```
ubuntu@ip-172-31-42-83:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        49G   23G   26G  48% /
```

Fig. 9. VICSORT Hard Disk Utilisation

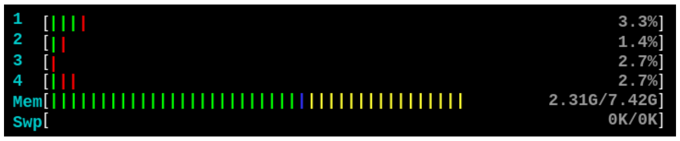


```
1  [                                    0.0%]
2  [                                    0.0%]
3  [|                                   0.7%]
4  [                                    0.0%]
Mem[||||||||||||||||||||||||||   1.07G/7.42G]
Swp[                                 0K/0K]
```

Fig. 10. Host Machine RAM utilisation just after boot

VI. VICSORT Contributions and Availability

This section summaries the contributions made by VICSORT as follows:



```
1  [||||                                                3.3%]
2  [||                                                  1.4%]
3  [|                                                   2.7%]
4  [|||                                                 2.7%]
Mem[||||||||||||||||||||||||||||||||||||||||||   2.31G/7.42G]
Swp[                                                 0K/0K]
```

Fig. 11. Host Machine RAM utilisation with VICSORT running

- Following successful boot of the host-compute, the initialisation script described in paragragh IV-Dc brings the testbed to an operational state in under 10 seconds.
- VICSORT shrinks the required RAM utilisation to run the testbed to about 1.2GB with a total of about 2.4GB being utilised by a host-compute with an operational VISCORT implementation. This is a reduction from the 5GB minimum RAM requirement in GRFICSv2.
- VICSORT shrinks the required hard disk space required by the host-compute to about 23GB for an operational implementation. This is a reduction from the 38.69GB required in GRFICSv2.
- VICSORT makes OS updates utilising more recent OS implementations than GRFICSv2.
- VICSORT provides an F/LOSS light-weight build that is easily deployable on both a Type 2 hypervisor or a Cloud implementation.
- VICSORT provides a detailed installation guide to support learners that wish to replicate and build upon this deployment.

VICSORT is available at *https://gitlab.com/ekisac10/vicsort* on GitLab. Included in this repository will be VICSORT's required setup files, testbed scripts and setup manual. A VirtualBox *.ova* file is also available to support quick implementation of the complete testbed on a VirtualBox VM.

VII. Conclusion and Future Work

This paper has introduced VICSORT – a light F/LOSS ICS testbed solution intended to be repeatable, scalable and easy to deploy. This is in an effort to lower the barriers to entry to ICS cybersecurity and provide a playground to further the ICS cybersecurity body of knowledge. This tool is licensed under the European Union Public Licence version 1.2 (EUPLv1.2) [26] and is currently in its first iteration (v1.0).

This paper wishes to acknowledge and recognise the work of the Georgia Institute of Technology and Fortiphyd Logic [2] in the development of the GRFICSv2 from which VICSORT builds upon.

For future works, it is intended that VICSORT will incorporate an Enterprise component as described by the Purdue model. The objective is to demonstrate the synergies between the Enterprise and Control zones as well as demonstrate how Information Technology (IT) cybersecurity affects Operational Technology (OT) and how IT and OT cybersecurity can complement each other.

References

[1] M. Conti, D. Donadel, and F. Turrin, "A Survey on Industrial Control System Testbeds and Datasets for Security Research," no. February, 2021. [Online]. Available: http://arxiv.org/abs/2102.05631

[2] D. Formby, M. Rad, and R. Beyah, "Lowering the Barriers to Industrial Control System Security with GRFICS," Georgia Institute of Technology and Fortiphyd Logic, Baltimore, MD, Tech. Rep., 2018. [Online]. Available: https://www.usenix.org/conference/ase18/presentation/formby

[3] C. Ekisa, D. Briain, and Y. Kavanagh, "An open-source testbed to visualise ics cybersecurity weaknesses and remediation strategies – a research agenda proposal," in *2021 32nd Irish Signals and Systems Conference (ISSC)*, 2021, pp. 1–6.

[4] L. Obregon and B. Filkins, "SANS Institute Information Security Reading Room Secure Architecture for Industrial Control Systems," Tech. Rep., 2021.

[5] Modbusorg, "MODBUS Application Protocol 1 1 b," Modbus.org, Tech. Rep., dec 2006. [Online]. Available: http://www.modbus-ida.org

[6] T. Alves and T. Morris, "OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research," *Computers and Security*, vol. 78, pp. 364–379, sep 2018.

[7] B. Genge, C. Siaterlis, I. Nai Fovino, and M. Masera, "A cyber-physical experimentation environment for the security analysis of networked industrial control systems," *Computers and Electrical Engineering*, vol. 38, no. 5, pp. 1146–1161, 2012.

[8] K. Mclaughlin and S. Sezer, "An Open Framework for Deploying Experimental SCADA Testbed Networks," *SCADA Cyber Security Research*, pp. 92–101, 2018. [Online]. Available: https://doi.org/10.14236/ewic/ICS2018.11

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

[10] M. Bertrand and T. Olivier, "Dissertation - Simulating Industrial Control Systems Using Mininet," 2017. [Online]. Available: https://dial.uclouvain.be/memoire/ucl/en/object/thesis%3A14706/datastream/PDF_01/view

[11] P. Maynard and K. Mclaughlin, "Ics interaction testbed: A platform for cyber-physical security research," 2019.

[12] R. Candell, K. Stouffer, and D. Anand, "A cybersecurity testbed for industrial control systems," 10 2014.

[13] E. Korkmaz, A. Dolgikh, M. Davis, and V. Skormin, "Ics security testbed with delay attack case study," 11 2016, pp. 283–288.

[14] J. Gardiner, B. Craggs, B. Green, and A. Rashid, "Oops i did it again: Further adventures in the land of ics security testbeds," 11 2019, pp. 75–86.

[15] D. Formby, "GitHub - djformby/GRFICS: Graphical Realism Framework for Industrial Control Simulations," 2018. [Online]. Available: https://github.com/djformby/GRFICS

[16] T. J. McAvov, "BASE CONTROL FOR THE TENNESSEE EASTMAN PROBLEM," *Computers Chem Engng*, vol. 18, no. 5, pp. 383–413, 1994.

[17] M. Sistemas, "Release ScadaBR 1.2 · ScadaBR/ScadaBR · GitHub." [Online]. Available: https://github.com/ScadaBR/ScadaBR/releases/tag/v1.2

[18] P. Kamal, "Intrusion Detection using pfSense - Open Source FREEBSD Firewall," Tech. Rep., 2014. [Online]. Available: https://www.academia.edu/17560234/INTRUSION_DETECTION_USING_PFSENSE_OPEN_SOURCE_FREEBSD_FIREWALL

[19] U. Technologies, "Unity Real-Time Development Platform — 3D, 2D VR AR Engine." [Online]. Available: https://unity.com/

[20] T. Alves, "Openplc editor," 2018. [Online]. Available: https://www.openplcproject.com/plcopen-editor/

[21] E. S. Fencing, "pfsense," 2021. [Online]. Available: https://www.pfsense.org/

[22] "The freebsd project." [Online]. Available: https://www.freebsd.org/

[23] lubuntu Meilix, "lubuntu – lightweight, fast, easier." [Online]. Available: https://lubuntu.net/

[24] J. J. Downs and E. F. Vogel, "A plant-wide industrial process problem control," vol. 17, pp. 245–255, 1993.

[25] D. Formby, "GitHub - Fortiphyd/GRFICSv2: Version 2 of the Graphical Realism Framework for Industrial Control Simulation (GRFICS)," 2020. [Online]. Available: https://github.com/Fortiphyd/GRFICSv2

[26] G. Oettinger, "European union public lisence (eupl)," *Official Journal of the European Union*, pp. 1–6, 5 2007.