# Lowering the Barriers to Industrial Control System Security with GRFICS

*David Formby*
*Georgia Institute of Technology and Fortiphyd Logic*
*djformby@gatech.edu*

*Milad Rad*
*Georgia Institute of Technology*
*ghiasirad.milad@gatech.edu*

*Raheem Beyah*
*Georgia Institute of Technology and Fortiphyd Logic*
*rbeyah@ece.gatech.edu*

## Abstract

Despite the abundance of free online resources and increased research into innovative educational techniques, the shortage of cybersecurity skills in the workforce continues. The skills gap in the specific area of industrial control system (ICS) security is even more dismal due to the higher barriers to entry raised by the exclusive use of expensive, proprietary hardware and software and the inherent dangers of manipulating real physical processes. To help beginners in ICS security overcome these barriers to entry we developed a graphical realism framework for industrial control simulations (GRFICS). GRFICS virtualizes entire ICS networks, from the operator interface down to realistic simulations of the physical process visualized in a 3D game engine. Using this framework, students can practice exploiting common ICS vulnerabilities and vividly see the physical impact in the visualization of the process. After gaining a better appreciation of the close relationship between the cyber and the physical worlds in ICS networks, students can then practice hardening the network against such attacks. This free and open-source framework can be used as the basis for formal classroom instruction, ICS-specific CTF competitions, or for independent study.

## 1 Introduction

Thanks to the power of virtualization, the barrier to entry for general cybersecurity is lower today than it has ever been. Anyone with a computer and Internet connection can download the intentionally vulnerable Metasploitable [8] virtual machine (VM), a Kali Linux VM filled with offensive security tools, and begin getting hands-on practice with common security problems all for free. For even more realistic practice and less than $100, multiple companies offer complete online vulnerable networks for students to hone their penetration testing skills. However, until now there has been no equivalent for the area of industrial control system (ICS) security for students to freely learn about the unique challenges faced when attacking and defending ICS networks. In order to get hands-on experience in ICS security, students typically must pay thousands of dollars for on-site training or for purchasing their own equipment and software to practice on. Even after those expenses it is difficult to truly appreciate the relationship between a cyber attack and a potential physical impact since it is too expensive and time-consuming to repeatedly cause kinetic damage to some physical process.

Therefore, in order to lower these barriers to entry for ICS security, we are releasing the first free and complete virtual ICS network [4] including every level from the human machine interface (HMI) in the control room, to the programmable logic controller (PLC) controlling the plant, down to a realistic simulation of the physical process itself visualized using the popular Unity 3D game engine.

The significant contributions of this research include:

- Conversion of the simplified Tennessee Eastman challenge process simulation into a more portable and accessible format
- Novel 3D visualization of a dynamic chemical process simulation to increase engagement and realism
- The most complex and complete virtualization of an ICS network to date, released free and open-source
- Modular framework for easy expansion or conversion to other physical processes and protocols

The remainder of this paper is organized as follows. Related work in ICS testbeds and virtualizing ICS networks is reviewed in Section 2 followed by a brief overview of ICS networks and the specific lessons in ICS security this framework intends to help teach in Section 3.1. Section 4 details the various pieces of the framework including the process simulation and visualization,

remote I/O modules, virtual PLC, and operator HMI. Finally, example offensive and defensive exercises are walked through in Section 5 followed by conclusions and future work in Section 6.

## 2 Related Work

Most of the related work into ICS testbeds has been focused on building real hardware testbeds primarily for the purpose of conducting high-fidelity research rather than for facilitating education. The most extensive examples have been produced by the Singapore University of Technology and Design where researchers have built a six-stage water treatment facility [27], water distribution network, and small scale electric power grid network[11]. The National SCADA Testbed [9] developed by the Department of Energy is another example of a large scale physical testbed including full scale electrical substations and transmission lines. Several other smaller physical testbeds have been built as well [23],but as useful as these physical testbeds are for research, they suffer from extreme scalability limitations for training new security practitioners. Therefore simulated ICS testbeds are necessary to truly lower the barrier to entry in a scalable manner.

Unfortunately, most of the research into simulated ICS testbeds has been severely limited in scale and fidelity. Several works [22] [29] [28] have simply used Omnet++ to examine basic attacks such as denial of service (DoS) on relatively simple physical processes. Others have implemented more detailed physical simulations [20] [25] and larger scale simulations of the power grid [19] but still suffer from scalability due to the requirements of proprietary software such as Simulink or cluster computing resource requirements.

One of the most significant barriers to high fidelity large scale virtualization of ICS networks was the lack of free virtualized alternatives to hardware programmable logic controllers (PLCs). Thankfully, a project began in 2014 to develop an open source PLC that can run on an open source hardware design or in pure software [17]. Expanding on this project the researchers have also begun developing virtual ICS networks using the OpenPLC[16]. However, the physical processes used in these virtual networks are still relatively small and require Simulink/Matlab, which limits their usability. Furthermore, the physical processes are only visualized in 2D human machine interfaces (HMIs) and line graphs that are not immersive and convincing. In summary, the work proposed here improves on previous work by providing a realistically complex physical simulation that is convincingly visualized in 3D graphics, provides all the necessary components for students to practice common ICS attacks and defenses, and is available completely free and open-source.

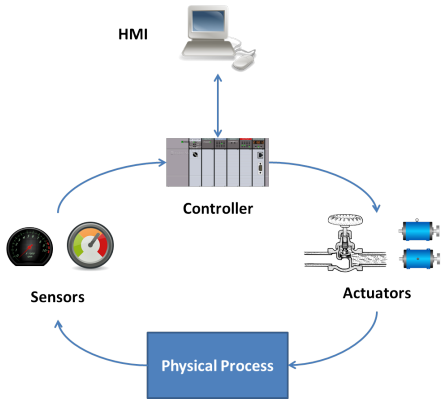


Figure 1: High level structure of ICS network

## 3 Background

To understand the GRFICS framework, we first must provide a brief overview of ICS terminology and basic structure, as well as the most important differences between ICS networks and IT networks.

### 3.1 Overview of ICS Networks

The term “industrial control system” is extremely broad and includes SCADA (supervisory control and data acquisition) systems used in power distribution, DCS (distributed control systems) used in oil refineries, and even building control systems. They all vary slightly in their architecture and specific devices, but they all share the same high-level structure illustrated in Figure 1, where a controller or controllers continuously read process measurements from sensors, report the measurements to some kind of human machine interface (HMI), and use the measurements with some combination of user input from the HMI and its own preprogrammed control logic to update the physical actuators, such as a valve or relay.

### 3.2 ICS Security Lessons

One of the causes for the skills gap in ICS security and the motivation for this work is the number of differences between traditional IT security and ICS security. In traditional security, the order of priorities is always given as confidentiality, integrity, and availability in that order. However, in ICS networks the priorities are essentially reversed. Confidentiality of the process measurements being sent over the local network at a power generation facility is the least of the operators’ worries. On the other hand, ensuring that the facility remains available to keep the lights on is their top priority. These differences in

priorities influence the design of both ICS devices and network architectures that lead to unique challenges in defending and attacking ICS networks that this framework aims to teach to a broader audience.

**ICS Endpoint Insecurity.** In IT security, there is a variety of tools and strategies for securing endpoints including frequent patching, host based firewalls, antivirus software, strong password enforcement, and support for secure communication protocols. In ICS security, due to the focus on availability and high uptime requirements, devices may only be patched once a year, replaced every 10-20 years, and are typically shipped with no passwords enabled by default and no password strength policy enforcement. Furthermore, due to the low powered hardware, devices are incapable of running antivirus software or even supporting authenticated network protocols. As a result, the only defenses a typical ICS endpoint has is the "security through obscurity" gained from running proprietary closed-source software and protocols. The tools provided in this work help teach students these lessons by having them practice exploiting data and command injection against unauthenticated protocols, exploiting buffer overflows in control system protocols, and brute forcing weak passwords.

**Importance of Network Defense.** Since the ICS endpoints themselves are essentially "insecure by design" and incapable of being upgraded, the security of an ICS network relies on having strong defense-in-depth built into the network. Specifically, the NIST Guide to ICS Security [31] stresses the importance of segmenting networks into enclaves, writing strong firewall rules, and using network intrusion detection systems. The framework proposed in this paper helps students learn these lessons by enabling them to practice writing firewall rules to segment the ICS network and writing intrusion detection rules to detect common ICS attacks.

**ICS Cyber Kill Chain.** Another important difference between ICS and IT networks is the end goals of the attacker and the number of steps necessary to reach them. In IT networks, the generally accepted "Cyber Kill Chain" involves reconnaissance, weaponization, delivery, exploitation, installation, command and control, and finally actions on the objective [24]. However, in the ICS world, the steps leading up to actions on the objective are only half the battle. The ICS Cyber Kill Chain [18] involves two stages, where the first stage closely resembles the original Cyber Kill Chain. However, in Stage 2 the attackers must develop an "exploit" of the physical system as well, on top of the series of exploits used on computing systems. In this physical exploit phase, attackers must understand the physical process at the victim facility, research the built-in safety checks and redundancies specific to that victim, and develop a strategy for bypassing those obstacles and achieving whatever their physical goal is. This paper helps teach this lesson by providing students with a nontrivial physical process simulation that they must study in order to design an attack that causes significant physical damage.

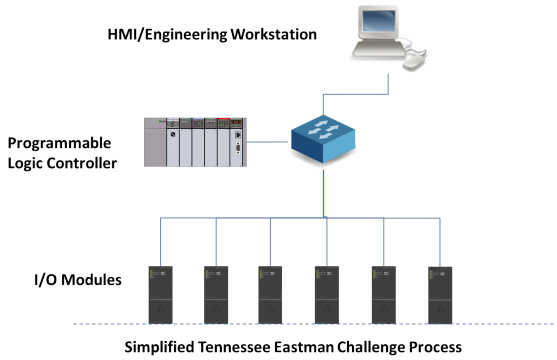


Figure 2: Network Diagram for Virtualized Network

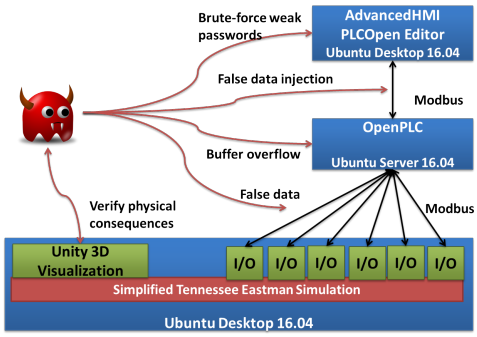


Figure 3: Architecture of GRFICS framework

## 4 GRFICS

The Graphical Realism Framework for Industrial Control Simulations (GRFICS) was designed to be modular to allow for easy customization and expansion. The ICS components including the HMI, PLC, and I/O modules are all connected using standard ICS network protocols, allowing for the PLC or HMI to be swapped out with high-fidelity real ICS devices. The physical process simulation is then tied to the 3D visualization and the I/O module layer using an intuitive JSON [5] based network API, documented in more detail in Section 4.1.2.

In this initial version of GRFICS, we are virtualizing a chemical process control network with a flat, unsegmented network architecture illustrated in Figure 2. Figure 3 illustrates how GRFICS virtualizes the whole control network with three virtual machines and allows users to practice common ICS attacks while verifying the physical impact in the 3D visualization. The remainder of this section describes each module in further detail.

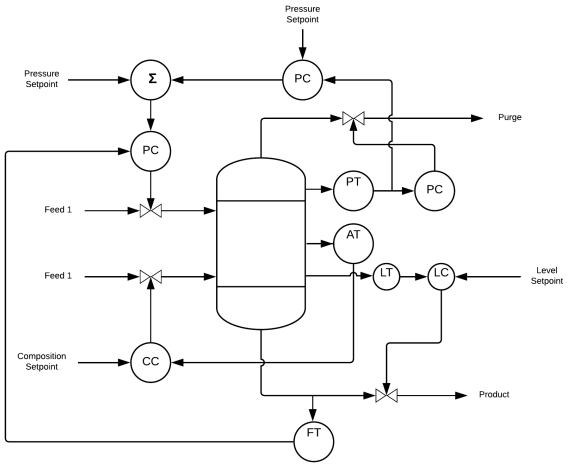


Figure 4: Piping & Instrumentation Diagram (P&ID) for the simplified Tennessee Eastman Challenge Process

## 4.1 Physical Process Simulation

The physical process simulation comprises the simulation backend, the simulation API, the 3D visualization, and the IO modules.

### 4.1.1 Simulation Backend

The simulation backend provides efficient, high-fidelity calculation of the current state of the simulated process. In this case, the process being simulated is a simplified form of the process control problem proposed by the Eastman Chemical Company in 1992 [21]. In the original paper, the researchers provided a Fortran code simulation of an industrial chemical process involving two exothermic reactions, two byproduct reactions, a total of 12 control valves to manipulate, and a total of 41 output measurements to monitor. The chemical reactions were nonlinear and optimal control of the process required keeping pressures and temperatures within safe ranges in the presence of disturbances while balancing the competing requirements to minimize costs and maximize efficiency.

In order to focus on a certain challenging aspect of the process, a researcher at the University of Washington released a simplified version of the process a year later in 1993[30]. This version simplified the process into one two-phase chemical reactor/separator illustrated in Figure 4 that included a total of four control valves to manipulate and ten output measurements to monitor. When designing a control system for this process, the reactor pressure must be kept at a safe level while maximizing the efficiency of the chemical reaction and minimizing the components being wasted through the purge valve. For GRFICS, we converted this simplified simulation from the original Fortran and Matlab into a more portable and standalone C++ program.

### 4.1.2 Simulation API

The simulation backend needed a simple and efficient way for reporting the process measurements to the 3D visualization and I/O modules, and receiving updates on the control valve positions. To achieve this, GRFICS communicates over TCP port 55555 using a simple, human readable JSON based API that can be extended for use with other process simulations.

```
{
    ''request'': ''read''
}
```

(a) JSON API Read Measurement Request

```
{
    ''process'':''simpleTE'',
    ''outputs'': {
        "f1_flow":1,
        "f2_flow":2,
        "purge_flow":3,
        "product_flow":4,
        "pressure":5,
        "liquid_level":6,
        "A_in_purge":7,
        "B_in_purge":8,
        "C_in_purge":9,
        "cost":10
    },
    "state": {
        "f1_valve_pos":1,
        "f2_valve_pos":2,
        "purge_valve_pos":3,
        "product_valve_pos":4
    }
}
```

(b) JSON API Read Measurement Response

Figure 5: JSON API Examples

For the 3D visualization or the I/O modules to obtain the current state of the chemical process, they simply send the JSON read request shown in Figure 5a. The simulation server responds with a JSON object (Figure 5b) containing the name of the process being simulated and the current values of all the output and state variables in the process. When the I/O modules receive a command to change a valve position, they send a JSON write request illustrated in Figure 6 to tell the simulation server to update the valve setpoints. The server then responds with another object like Figure 5b showing the new state of the process based on the new valve positions.

### 4.1.3 3D Visualization

One of the primary weaknesses of all previous work on ICS network virtualization is the lack of engaging realism for the physical process. To date all other attempts have simply used basic line graphs of critical process measurements, or the same human machine interfaces

```
{
    "request":"write",
    "data": {
        "inputs": {
            "f1_valve_sp":1,
            "f2_valve_sp":2,
            "purge_valve_sp":3,
            "product_valve_sp":4,
        }
    }
}
```

Figure 6: JSON API Update Inputs Example

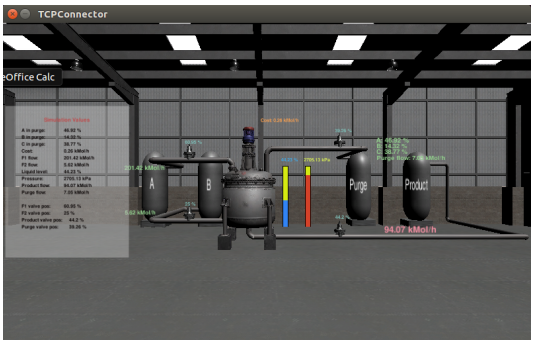


Figure 7: "Normal" Operation

(HMI) that operators already use based on simplified 2D graphics. There was no easy and interesting way for students to see the effects of injecting false data into the network and blinding the operator's HMI or for students to appreciate the physical consequences of their attacks.

Inspired by the past successes with flight simulators and military training simulators[2], the GRFICS framework uses a 3D game engine to visualize the true state of the physical process with more engaging realism. For the simplified Tennessee Eastman challenge process being simulated here, GRFICS uses the Unity 3D game engine, which is used by more mobile game developers than any other third party game engine software[6].

In Unity 3D, developers can easily create games by dragging and dropping 3D objects into the current scene and write C# or Javascript code to describe how the object behaves and is updated each time a frame is rendered. For this version of GRFICS, we purchased premade 3D models of a warehouse environment, industrial reactor, pipes, and valves and used them as building blocks to create a realistic chemical manufacturing facility. The Unity application connects to the simulation backend server, requesting updated measurements for each frame, and displaying them over each component and in a side panel summary.

As students practice attacking the virtual chemical process control network, they can compare the true state of the process in Figure 7 side by side with the HMI screen that the operator at the victim facility would see, discussed in more detail in Section 4.3. When a student

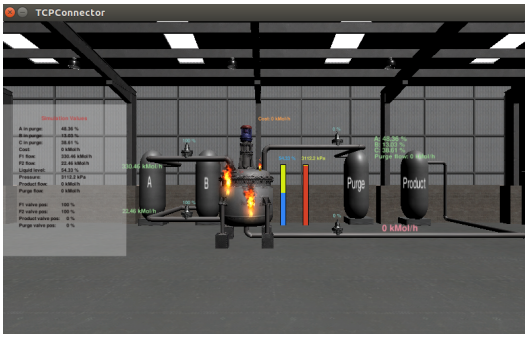


Figure 8: Pressure Limits Exceeded

who is attacking the system succeeds in forcing the pressure in the reactor vessel to exceed the safety limit of 3200 kilopascals, an explosion effect plays on top of the reactor vessel followed by fire effects, illustrated in Figure 8.

#### 4.1.4 I/O Modules

The I/O modules are the pieces that tie the process simulation to the virtual ICS network. On the same VM as the simulation backend and visualization, simple Modbus servers are created that listen on many different IP address aliases to appear as multiple devices on the network. Each Modbus server continuously polls the simulation backend for current measurements and reports different measurements over Modbus. For example, one Modbus I/O server reports the position of the product valve, the flow rate through it, and accepts Modbus write commands to update the position of the valve.

Since Modbus, and virtually all other ICS network protocols, are unauthenticated, students can practice injecting commands directly to the I/O modules, launching man-in-the-middle attacks to report false data, and designing firewall rules and intrusion detection rules to protect the insecure-by-design I/O. Due to the low-fidelity simplifications, more advanced attacks like buffer overflows and firmware reverse engineering are not supported at this level.

### 4.2 Programmable Logic Controller

The programmable logic controller (PLC) component of the framework continuously polls the Modbus IO for the current state of the process, executes a simple control logic program to determine necessary control actions, and sends the updated control valve positions to the Modbus IO. Additionally, the PLC accepts control commands from the HMI and responds to read requests for process measurements. In this initial version, we use the OpenPLC project [17] for the PLC component to keep the en-

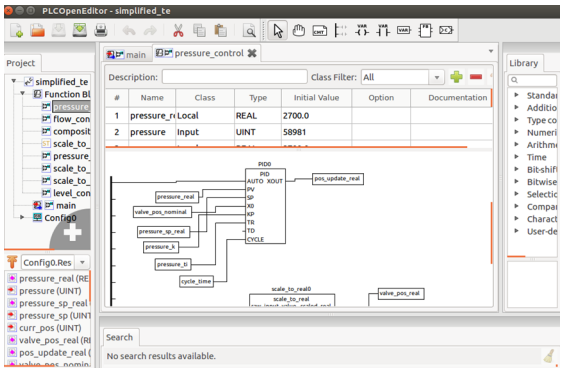


Figure 9: PLCOpen Editor

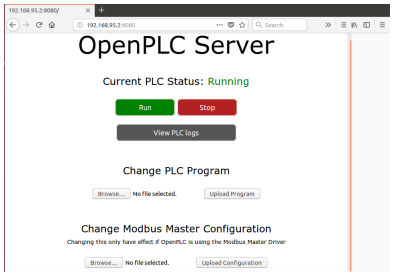


Figure 10: OpenPLC Web Interface

tire framework free and open source. Additionally, we replace the up-to-date version of the libmodbus library that the PLC uses with an older version vulnerable to buffer overflow attacks. In higher fidelity virtual ICS networks, either hardware PLCs can be used in the loop or proprietary software PLCs running in VMs can be used.

The PLC program itself implements the multi-loop control strategy proposed in the original paper [30] using a combination of PID (proportional-integral-derivative) controllers for the different process measurements. To program PLCs in the real world, engineers must develop the PLC program on a separate engineering workstation and then upload the program to the PLC. In this work, we use the PLCOpen Editor software 9 to create the program and then upload the program using OpenPLC's web interface, Figure 10.

## 4.3 Human Machine Interface

The last piece of the GRFICS framework is a combined human machine interface (HMI) and engineering workstation. HMIs are typically 2D graphical interfaces for operators of the ICS facility to monitor key measurements of the process and occasionally interact with the process using Start/Stop buttons or updating setpoints. In this version of GRFICS, we used an open-source HMI product under active development called AdvancedHMI [1].

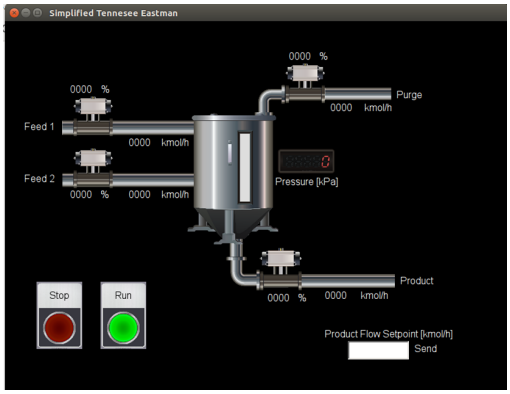


Figure 11: Operator Human Machine Interface (HMI)

Using the AdvancedHMI design environment, we developed an HMI (Figure 11) for the simplified Tennessee Eastman Chemical process being simulated. Operators can use the Stop button to send the process into a "safe" state, restart it with the "Run" button, and update the product flow setpoint with the text box entry. Note that in real world facilities startup and emergency shutdown procedures are usually more complicated to ensure a safe and gradual change of state.

# 5 Example Exercises

To encourage adoption of GRFICS for learning and teaching ICS security, we now provide a brief overview of suggested offensive and defensive exercises. In these exercises, we use several VirtualBox VMs running Ubuntu Desktop for the simulation VM and HMI VM, and Ubuntu server for the PLC VM. We configured a host-only networking adapter to create a virtual network where all devices can communicate with one another, and the host machine can capture the network traffic for review.

## 5.1 Offense

For the offensive exercises, we assume the attacker has successfully phished an employee on the corporate network, compromised their machine, and is attempting to use it to pivot into a poorly protected control network to perform reconnaissance and ultimately cause physical damage.

### 5.1.1 Password Cracking

To allow for easy maintenance on the control system by third party contractors or for remote monitoring of the process, many facilities have remote access enabled to machines on the plant floor. Unfortunately, this is not

always securely implemented with two-factor authentication or even strong password policies. To gain the initial foothold on the control system network, students can practice cracking weak SSH passwords on the HMI VM using tools such as Ncrack or Hydra.

#### 5.1.2 ICS Reconnaissance

Once they have cracked the weak SSH password and gained access to the combined HMI and engineering workstation, students can begin reconnaissance on the physical process of the facility they are attacking. This can include exploring the machine to look for process diagrams (Figure 4), PLC program files (Figure 9), and the HMI screen (Figure 11). Students can also perform standard network scans with Nmap [10] to see how many devices are on the control network, perform man-in-the-middle (MITM) attacks with Ettercap [3] to study how they are communicating, and examine the devices deeper with free Modbus scanners. Note however that in the real world some legacy devices are so fragile that a standard Nmap scan can overwhelm them with bandwidth and concurrent TCP connections causing them to crash.

#### 5.1.3 Modbus Command Injection

After students have identified all Modbus devices, gained a basic understanding of the physical process, and determined the mapping between addresses and physical measurements, they can begin attacking the physical process. One of the most straightforward attacks takes advantage of the unauthenticated Modbus protocol and simply sends direct Modbus commands to the IO devices or the PLC. Students can either practice writing their own scripts to send the malicious commands, or simply use an existing tool like QModMaster [13].

However, this simple approach has a few drawbacks. First, injecting commands alone would likely be detected by the operators monitoring the HMI. To be successful the attacker would also need to be performing a MITM attack blinding the HMI and/or the PLC. Second, depending on the safety checks built in to the PLC logic, it may be difficult or impossible to achieve significant damage without also sending false data to the PLC. Finally, these attacks could be easily detected and prevented with basic intrusion detection systems and network segmentation.

#### 5.1.4 PLC Program Modifications

Another approach to achieving a successful physical attack on the system would be to modify the PLC programming, as Stuxnet[26] and Triton[15] did. Unfortunately, most PLCs have weak access control with password authentication disabled by default and no password strength policies, making it trivial to log in and reprogram the PLC using the legitimate engineering software. OpenPLC is no exception to this rule, so students can practice modifying the program in the PLCOpen Editor and uploading new malicious programs to the OpenPLC's web interface.

#### 5.1.5 Modbus Buffer Overflow

Finally, the abundance of legacy devices and slow patching cycle in ICS networks means there is a high chance of protocol vulnerabilities in the end devices. Buffer overflow vulnerabilities are one of the most common and typically arise from accepting user input strings without proper bounds checking. Attackers crafting an exploit for these vulnerabilities have to ensure that their payload does not contain any bad characters (such as whitespace) that would terminate the string early. However, ICS protocols are primarily machine-to-machine communication and instead of accepting human readable strings they exchange raw binary information, which makes generating malicious payloads even easier.

Even in the open source community, vulnerabilities can remain unpatched for far too long. For example, a buffer overflow vulnerability was publicly reported in the libmodbus library [7], a free open-source Modbus implementation, in September 2011 and was not fixed until two years later. The OpenPLC project uses an updated and patched version of this library, but for GRFICS we reverted the library back to a older vulnerable version, with relevant code snippets of the vulnerability illustrated in Figure 12.

The specific vulnerability stems from the library allocating a fixed amount of memory for every response, and not performing the necessary bounds checking to ensure that it doesn't try to respond with too much data. This means that sending a read request for data longer than 260 bytes would result in the library overflowing the buffer and overwriting the return address with whatever data had been requested. In order for attackers to control this data, and thereby control the return address, they must use a lesser known Modbus function code that first writes arbitrary data to a specific address range and then immediately reads data from a specified address range. With the right combination of write and read address ranges, attackers can craft a reliable exploit that achieves arbitrary code execution. This exercise provides students with a way to practice and learn about buffer overflow vulnerabilities in the context of ICS networks and then use the exploit to attempt to cause physical damage to the process.

```
#define MAX_MESSAGE_LENGTH 260

...

int modbus_reply(modbus_t *ctx, const uint8_t *req,
                 int req_length, modbus_mapping_t *mb_mapping)
{
...
    uint8_t rsp[MAX_MESSAGE_LENGTH];
...
    case _FC_WRITE_AND_READ_REGISTERS: {
...
            for (i = address; i < address + nb; i++) {
                rsp[rsp_length++] = mb_mapping->tab_registers[i] >> 8;
                rsp[rsp_length++] = mb_mapping->tab_registers[i] & 0xFF;
            }
```

Figure 12: Code snippets of libmodbus buffer overflow vulnerability

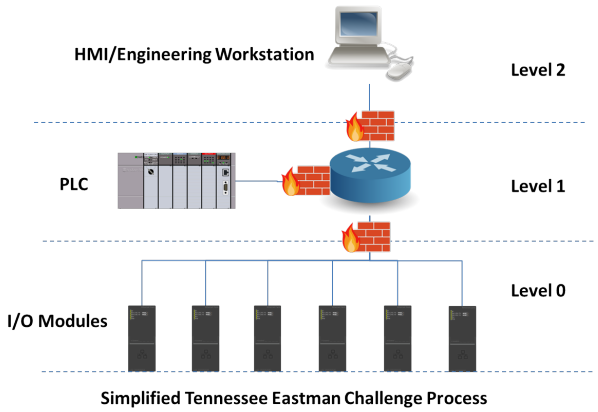


Figure 13: Segmented network architecture according to Purdue Reference Model

## 5.2 Defense

In this section we provide a brief walkthrough of defenses that can be deployed to prevent and mitigate the attacks explained in the previous section. For these defensive exercises, we assume that the defending facility has an extremely limited budget and can only deploy a single unified threat management(UTM) appliance/router on the control network to implement firewall and intrusion detection rules. Again, to keep GRFICS completely free and open-source we use the OpenWrt project [12], an embedded Linux OS for routers.

### 5.2.1 Network Segmentation

Network segmentation is one of the first recommendations for increasing ICS network security and involves segregating network nodes into zones based on functionality and trust level, only allowing communication between zones or nodes that absolutely need it. This makes it significantly harder for attackers to expand their foothold throughout the network and to make direct attacks on the physical process. For these exercises, students can practice separating the nodes in the GRFICS framework into subnets based on the hierarchies described in the ISA-95 standard and illustrated in Figure 13. Firewall rules can be applied using raw iptables rules or OpenWrt's graphical interface to whitelist nodes and protocols.

### 5.2.2 Intrusion Detection and Prevention

Network segmentation makes it significantly harder for an attacker to breach the control network, but not impossible. To prevent the attacker from causing any physical harm, defenders need to be able to quickly detect the intrusion and can even implement automated defenses to prevent some of the damage.

While there are a variety of companies selling ICS-specific intrusion detection (IDS) and prevention systems (IPS), Snort [14] is a popular free and open-source solution for students to practice learning what such systems are capable of. Using Snort, students can write rules to detect the various attacks they practiced in Section 5.1. For example, Snort rules can be written to alert operators of attackers performing reconnaissance by detecting suspected SSH password guessing, network scanning, and Modbus scanning. With a little reverse engineering work, students can also write rules to detect when new programs are uploaded to the PLC and when the PLC is started and stopped. Finally, Snort provides intrusion prevention functionality to actively block certain packets if it matches certain rules. Using Snort's intrusion prevention functionality, students can write rules to block the vulnerable Modbus function code entirely or only block it if an exploitation attempt is detected.

## 6 Conclusions and Future Work

The shortage of skills in ICS security lags even further behind traditional cybersecurity due to much higher barriers to entry. While there are plenty of options for hands-on practice with network penetration testing and defense, none of them help teach ICS-specific skills. To address this need, we developed the GRFICS framework for virtualizing entire ICS networks including realistic physical processes simulations, which is visualized using the Unity 3D game engine to increase realism and engagement. We then outlined example exercises students can use to get hands-on experience both attacking and defending ICS networks and physical processes. In future work we will continue developing ICS simulations and visualizations for a broader range of applications, such as water treatment facilities and discrete manufacturing, and expand the virtual network with other common components, including process historians.

## References


[1] Advancedhmi. https://www.advancedhmi.com/.

[2] America's army. https://www.americasarmy.com/press.

[3] Ettercap home page.

[4] A graphical realism framework for industrial control simulations. https://github.com/djformby/GRFICS.

[5] Introducing json. https://www.json.org/.

[6] The leading global game industry software. https://unity3d.com/public-relations.

[7] libmodbus.org. http://libmodbus.org/.

[8] Metasploitable 2 exploitability guide. https://metasploit.help.rapid7.com/docs/metasploitable-2-exploitability-guide.

[9] National scada testbed. https://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf.

[10] Nmap: the network mapper. https://nmap.org/.

[11] A one-of-a-kind garden of testbeds. https://itrust.sutd.edu.sg/testbeds/overview/.

[12] Openwrt project: About the openwrt/lede project. https://openwrt.org/about.

[13] Qmodmaster. https://sourceforge.net/projects/qmodmaster/.

[14] What is snort? https://www.snort.org/faq/what-is-snort.

[15] Mar-17-352-01 hatmansafety system targeted malware. https://ics-cert.us-cert.gov/sites/default/files/documents/MAR-17-352-01\%20HatMan\\\%E2\%80\%94Safety\%20System\%20Targeted\%20Malware_S508C.pdf, 2017.

[16] ALVES, T., DAS, R., AND MORRIS, T. Virtualization of industrial control system testbeds for cybersecurity. In *Proceedings of the 2Nd Annual Industrial Control System Security Workshop* (New York, NY, USA, 2016), ICSS '16, ACM, pp. 10–14.

[17] ALVES, T. R., BURATTO, M., DE SOUZA, F. M., AND RODRIGUES, T. V. Openplc: An open source alternative to automation. In *IEEE Global Humanitarian Technology Conference (GHTC 2014)* (Oct 2014), pp. 585–589.

[18] ASSANTE, M. J., AND LEE, R. M. The industrial control system cyber kill chain. https://www.sans.org/reading-room/whitepapers/ICS/industrial-control-system-cyber-kill-chain-36297.

[19] BERGMAN, D. C., JIN, D., NICOL, D. M., AND YARDLEY, T. The virtual power system testbed and inter-testbed integration. In *Proceedings of the 2Nd Conference on Cyber Security Experimentation and Test* (Berkeley, CA, USA, 2009), CSET'09, USENIX Association, pp. 5–5.

[20] CHABUKSWAR, R., SINPOLI, B., KARSAI, G., GIANI, A., NEEMA, H., AND DAVIS, A. Simulation of network attacks on scada systems. In *First Workshop on Secure Control Systems* (2010).

[21] DOWNS, J., AND VOGEL, E. A plant-wide industrial process control problem. *Computers & Chemical Engineering 17*, 3 (1993), 245 – 255. Industrial challenge problems in process control.

[22] GHALEB, A., ZHIOUA, S., AND ALMULHEM, A. Scada-sst: a scada security testbed. In *2016 World Congress on Industrial Control Systems Security (WCICSS)* (Dec 2016), pp. 1–6.

[23] GREEN, B., LEE, A., ANTROBUS, R., ROEDIG, U., HUTCHISON, D., AND RASHID, A. Pains, gains and plcs: Ten lessons from building an industrial control systems testbed for security research. In *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17)* (Vancouver, BC, 2017), USENIX Association.

[24] HUTCHINS, E. M., CLOPPER, M. J., AND AMIN, R. M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. https://www.lockheedmartin.com/\\content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf.

[25] KROTOFIL, M. Rocking the pocket book: Hacking chemical plants for competition and extortion. https://www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf, 2015.

[26] LANGNER, R. To kill a centrifuge. http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf, November 2013.

[27] MATHUR, A. P., AND TIPPENHAUER, N. O. Swat: a water treatment testbed for research and training on ics security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)* (April 2016), pp. 31–36.

[28] QUEIROZ, C., MAHMOOD, A., HU, J., TARI, Z., AND YU, X. Building a scada security testbed. In *2009 Third International Conference on Network and System Security* (Oct 2009), pp. 357–364.

[29] QUEIROZ, C., MAHMOOD, A., AND TARI, Z. Scadasim:a framework for building scada simulations. *IEEE Transactions on Smart Grid 2*, 4 (Dec 2011), 589–597.

[30] RICKER, N. L. Model predictive control of a continuous, nonlinear, two-phase reactor. *Journal of Process Control 3*, 2 (1993), 109 – 123.

[31] STOUFFER, K., PILLITTERI, V., LIGHTMAN, S., ABRAMS, M., AND HAHN, A. Guide to industrial control systems (ics) security. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-82r2.pdf.