Design and Implementation of a Cyber Physical Testbed for Security Training

Paul Pfister, Mathew L. Wymore, Doug Jacobson, and Daji Qiao
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA
{ppfister, mlwymore, dougj, daji}@iastate.edu

Abstract

This paper describes the work of introducing a CPS (Cyber Physical System) extension to our Internet event simulator at Iowa State University, known as ISEAGE, for Cyber Defense Competitions (CDCs). The CPS extension consists of a virtual control system to interface with CPS devices, a human machine interface (HMI) to access and control the devices, a virtual world to simulate the physical effects of the devices, and a backend to support the use of the CPS component in CDCs. These components communicate with each other using the Open Platform Communications standard. A CPS-CDC scenario is developed where participants are tasked with defending two CPS networks representing water and power utilities. To enhance the experience for participants, we also 3D-print a model of the virtual city served by these utilities. The 3D city reflects the state of the competition via LEDs that report the availability of services. The design of our CPS-CDC is highly modular, supporting any number of different CPS devices and systems, and can be adapted to a wide set of possible CPS scenarios.

1 Introduction

Cyber defense competitions (CDCs) were first introduced as a way to give participants key insights into the methods and mindset used by attackers against networked systems. Following the "know your enemy" form of thinking, a CDC educates the participants about what network defense is like in the real world. This gives them a more natural intuition for defensive strategies when they are put to test in the field. It also helps them to recognize and understand the vulnerabilities they encounter with the systems they are tasked to administrate, and to develop skills to identify potential attacks.

Traditionally, CDCs focus on defending networks in "cyberspace," and particularly on protecting the integrity, confidentiality and availability of data. As the Internet evolves, more and more physical systems in the real world become connected and thus can be accessed and controlled from the Internet. For example, the power that comes to our homes, the water from our faucet, the traffic lights that coordinate our driving, and many more, are now directly controlled by computers, and many are accessible from the Internet. These systems are often referred to as Cyber Physical Systems (CPS).

The dangers posed to CPS are growing at alarming rates due to the increasing rates at which they are being networked into the Internet. The recent attack on the Ukrainian power grid is likely to be the first of many examples, where hackers managed to disable a number of substations, leaving 230,000 residents without power [1]. However, there has been a lack of CDCs that simulate CPS systems and behaviors. This motivates us to design and implement a CPS testbed to complement our Internet event simulator at Iowa State University, known as ISEAGE [2] (Internet-Scale Event and Attack Generation Environment), for CDC and security training purposes.

1.1 Internet Event Simulators and CDC

There are a few well-known Internet event simulators and testbeds that have been used for CDCs, such as Emulab [3]: "a configurable Internet emulator in a room;" DETER [4]: "an evolving infrastructure—facilities, tools, and processes—to provide a national resource for experimentation in cyber security;" and PlanetLab [5]: "a safe and secure environment for testing and operating peer-to-peer algorithms and monitoring their activities." To the best of our knowledge, none of these has a CPS component and CPS-related security issues and scenarios cannot be included in CDCs that use these systems for security training. Our Internet event simulator at Iowa State University is known as ISEAGE, which is a controlled environment that allows real attacks to be played out against the students' networks and demonstrates to them real world security concepts. Over the years, we have hosted many CDCs on ISEAGE, but our traditional CDC framework does not have a CPS component either.

1.2 Cyber Physical Systems (CPS)

Cyber physical systems (CPS) are networks of software-managed devices that interface with mechanical components and influence/monitor the physical world via sets of inputs and outputs. This relationship between software and hardware is often referred to as *computation* and *process*. Computation controls the process and adapts to events reported from the hardware [6]. Because these devices are not generally meant for interfacing directly, they are most commonly controlled by algorithms and supervised remotely by other systems.

In large facilities, workers need a way to monitor and control the activities of CPS. For this task, supervisory control and data acquisition (SCADA) software is deployed. SCADA is capable of communicating with the underlying control systems while simultaneously providing an interface for the workers to monitor and govern those systems. SCADA is a means of management. Traditionally, the design of CPS has

not focused on security [7]. Instead, it is up to the administrators of the facilities to segment the networks to ensure the systems are only accessible by authorized users. Convenience or necessity, however, insists that many of the systems be remotely accessible from the Internet, which further complicates the goal of secure network segmentation.

1.3 Our Contributions

A CDC scenario that integrates a CPS element would benefit students, both upcoming and current administrators, and the general public at large. CDCs provide a simulated environment for participants to test their defensive strategies with real-time feedback. Broadening the scope of a CDC to include both cyberspace and cyber-physical elements would expand the coverage to encompass a wider scope of threats facing the modern world. A CPS-CDC needs to include physical-world elements, since the dangers faced by these systems could have tangible and potentially severe consequences.

We have designed and implemented a CPS component, as an addition to our ISEAGE Internet event simulator and our traditional CDC framework. Leveraging the Open Platform Communications (OPC) standard, we design a generic virtual control system (VCS) layer that can integrate any CPS device or system into our CPS extension. To demonstrate our CPS extension, we introduce two CPS scenarios, a water utility and a power utility, into our CDC. We also use a 3D-printed city model to display the status of CPS devices and enhance the CPS-CDC experience for participants.

2 Cyber Defense Competition (CDC)

In our CDC, participants are divided into the following teams:

Blue Teams Blue teams are the competitors of the CDC. During the competition, their role is to defend their systems and react to any unauthorized activities. They also must ensure that their services are available at all times and respond to any issues communicated by their users.

Red Team There is a single red team. The red team plays the role of attacker during the competition, acting like a malicious adversary. The red team members are volunteers, generally from sponsoring businesses.

Green Team There is a single green team. The green team acts as the user base for the blue team networks.

White Team There is a single white team. They are the administrators of the competition, and are responsible for coordinating activities, managing the testbed, and scoring the blue teams.

For the remainder of this document, we refer to CDC networks (enabled by the Internet event simulator) by the color of the team that owns the network; e.g., a "blue network" refers to a network owned and defended by a blue team, whereas a "white network" refers to an administrative network that blue teams cannot access.

Each CDC features a unique scenario. When the scenario is distributed, the blue teams are given a set of virtual machines (VMs) that they use to create their infrastructure. These VMs generally come with pre-deployed services for the scenario. Example services include Active Directory, remote desktop capabilities, and a content management service. But these VMs also come with security loopholes, which are represented by the strategically-placed special files, called *flags*, in the VMs. During the competition, the red team tries to obtain or modify the flags as proof that they have compromised the blue networks, while the blue teams try to fix security weaknesses in order to protect their flags.

Throughout the competition, extraneous events called *anomalies* are generated that the blue teams must also address. Anomalies force blue teams to divide their attention, make decisions in real-time, and adopt both reactive and proactive security strategies. When the competition ends, the blue teams are scored and ranked depending on anomalies solved, flags defended, and service availability. In the next section, we discuss how we extend this basic CDC framework to support a CPS component.

3 CPS-CDC: Design

As shown in Fig. 1, our design for a CPS extension to an Internet event simulator consists of four main parts. Two of them, CPS devices and the human-machine interface (HMI), are implemented in the blue networks. The other two, the virtual world and backend, are implemented in the white network, outside the control of the blue teams.

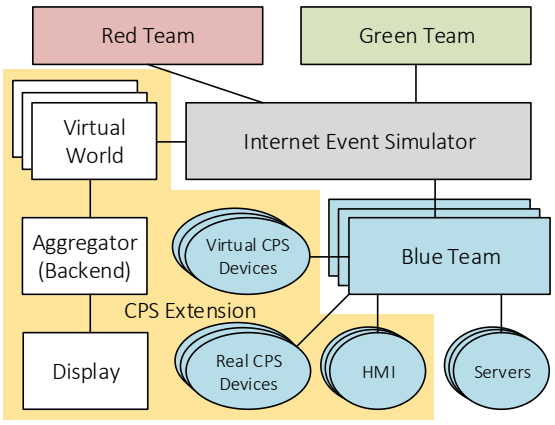

Figure 1: Overview of the CPS-CDC Design. The yellow shaded background indicates the components of the CPS extension.

3.1 CPS Devices

In a CPS-CDC, each blue team should be assigned multiple CPS devices to defend, but what will the devices represent, and how will they be implemented? We want to design a generic CPS device framework that can be applied to many different CPS-CDC scenarios, but such a design is challenging. There are many types of CPS devices, and from a networking perspective, many different network stacks and protocols that could be used. While some protocols are open standards, many are proprietary and dependent on the hardware being simulated. In addition, devices can operate at different layers of the protocol stack, further complicating the definition of a generic CPS device structure.

To solve this problem, we define a layer of abstraction that sits on top of CPS systems (either individual devices or multiple devices in a single system). This layer of abstraction, which we call the *virtual control system* (VCS), provides a universal interface to read and write data from CPS systems. This allows us to integrate any type of CPS system, whether virtual or physical hardware, by simply implementing the VCS layer for that system. With VCS at the top, any underlying systems and protocols can be supported. For reference, in Fig. 2, we present a typical CPS stack, composed of the following layers:

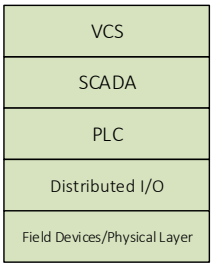

Figure 2: An example CPS stack. With VCS at the top, any lower layers can be supported in our CPS extension.

Field Devices/Physical Layer Devices at this layer are sensors or actuators that directly interact with the physical world. Examples include pumps, relays, switches, motors, etc.

Distributed I/O Devices at this layer translate the analog or digital signals of the field devices into protocols the PLCs (discussed next) can understand [8]. For example, a device may read in analog signals from a pump and translate those signals to Ethernet frames to relay to the PLC.

PLC A PLC (Programmable Logic Controller) is a special type of embedded system that coordinates and monitors the activity of field devices or communicates with the distributed I/O. A PLC is different from a standard computer in that it generally executes a single program and follows a linear execution flow [9]. A PLC may also have an operating system (typically a real-time operating system such as VxWorks). PLCs often run services like Telnet that make them remotely accessible. Using an Internet device search engine like Shodan, it is possible to discover PLCs directly connected to the Internet [10].

SCADA SCADA (supervisory control and data acquisition) systems are the traditional upper layer for CPS systems. In SCADA terminology, the system being monitored and controlled is the *process*. The SCADA system is an umbrella system responsible for collecting and monitoring aggregate data about the process and presenting that information to system administrators via a human machine interface (HMI). Operators can read data and make changes to the process through the HMI. SCADA systems are also responsible for alerting operators of any erroneous or critical conditions that may occur in the lower levels of the network [11]. There are numerous SCADA protocols, some of which are proprietary and specific to a vendor and others which are open source.

VCS As previously discussed, the top layer of our stack is our virtual control system, which provides a unified interface to CPS systems. We define the VCS layer with the help of an existing standard, Open Platform Communications (OPC, previously OLE for Process Control). OPC's purpose is to provide a unifying abstraction for a variety of other protocols, such as SCADA and PLC protocols. Thus, it aligns nicely with our goal for VCS.

OPC defines a set of commands that can be translated to the commands understood by specific systems, such as a SCADA protocol. There are several published forms of OPC, including OPC-DA, OPC-UA and OPC-XML. OPC-DA (Data Access) is sometimes referred to as Classic OPC. It relies on Microsoft's COM interface and is, therefore, platform-dependent. OPC-XML is platform-independent and uses a published XML format to communicate over the network [12]. OPC-UA (unifying architecture) is the newest of the standards and is also device-independent.

Since we desire to support a variety of devices, we select OPC XML/UA as the foundation for our VCS. The XML format provides the advantage of being human-readable. Additionally, OPC-UA is generally represented as a tree, which allows for easy generation of state machines. Finally, there are open source libraries that implement OPC-XML/UA, providing a solid foundation for our VCS layer. We use PyOPC, an open source library developed in the Python programming language [13] and later Python OPC-UA [14].

We use OPC as the basis for communication between the components of our CPS extension. OPC uses a client/server communications model, and Fig. 3 shows how OPC clients and servers are chained together in our framework to create the infrastructure for a CPS-CDC. As shown in the figure, our VCS layer is built around an OPC server and client pair. Both an OPC server and an OPC client are used in the VCS layer in order to connect each CPS system to both the HMI and the virtual world host, described in the following sections.

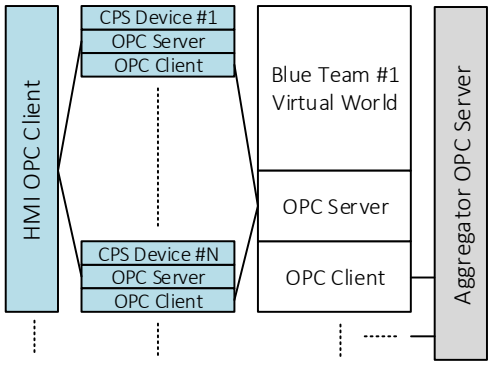

Figure 3: The OPC client-server chain used to connect the components of our CPS extension.

The data provided by a CPS device is a collection of *tags*. A tag is a SCADA version of a variable, meaning a value coming from a PLC. The tags represent the state of the CPS device and its interactions with the physical world. When other components of the CPS extension query a CPS device using OPC, the VCS responds with the appropriate tags. These tags can be populated by either physical hardware, or by a simulated, fully-virtual device.

The VCS, through virtue of its abstraction of the lower layers, can be implemented in any environment. However, we wish to keep the VCS as representative of an embedded system as possible. Therefore, we implement VCS using Buildroot [15], a framework to create tiny embedded Linux systems. With Buildroot, we create system images with a stripped set of userland tools and a minimal set of features. This makes the environment difficult to change or upgrade, as in a real embedded system, encouraging blue teams to use more realistic defense strategies.

The use of Buildroot also reduces the resources the CPS devices need to function, which is important due to the volume of virtual CPS devices distributed for a CPS-CDC. For example, in a recent competition, we distributed 11 virtual control devices per team, which can potentially be difficult to scale to larger competitions. Using Buildroot, we were able to limit the size of virtual CPS devices to about 150 megabytes. In addition, these devices need only a single CPU core and less than 512 megabytes of memory.

3.2 HMI

Each blue team in our CPS-CDC will be presented with a human-machine interface (HMI) that allows them to monitor their CPS devices and adjust the process those devices control. This HMI takes the place of the interface to the SCADA system that an operator would normally have. Because we use OPC in the VCS layer, we can easily design an abstracted, virtual HMI based around an OPC client. This client connects to the OPC server on the virtual control system for each device, as shown in Fig. 3. This allows the HMI to display information for each device, and to control those devices.

Because the HMI and VCS are both in scope for Red Team, we also create a scenario where Red Team may be capable of modifying values in the VCS while simultaneously reporting erroneous values to the HMI. In this manner we can model real world attacks where the system Administrators may not be privy to an attack on the physical environment, forcing students to take other measures to ensure the stability of their CPS network.

3.3 Virtual World

A key difference between our CPS extension and the base Internet event simulator is that our extension simulates interactions with the physical world. To do this, we design a *virtual world*, a simulated version of the physical world of the CPS-CDC scenario.

We implement the virtual world as a standalone *virtual world host*. A virtual world host tracks and represents the state of a single blue team's virtual world, with one virtual world host for each blue team. As shown in Fig. 1, the virtual world hosts reside in a white network, outside blue teams' control. Each CPS device (discussed in Section 3.1) has a link to the owning team's virtual world host. The virtual world host collects data from these CPS devices and computes how the devices' states influence the simulated virtual world. The CPS devices also read data back from the virtual world host, allowing them to adjust their state to reflect changes in the virtual world. This allows, for instance, for cascading failures among CPS devices.

The communication between the devices and the virtual world host is done with OPC. As shown in Fig. 3, the virtual world host acts as an OPC server, and the OPC clients in the VCS layer of the devices talk to this server. We implement the logic of the virtual world using *service scanner scripts* that run on the virtual world host. These scripts monitor the state of the CPS devices and maintain consistency in the virtual world and among the CPS devices.

3.4 Backend

The final component of our CPS extension is the *backend*, which monitors the virtual worlds of the blue teams, scores the CPS component of the CDC, and displays the states of the virtual worlds and the scores for each team. The backend is divided into three main components:

Virtual World Aggregator The *virtual world aggregator* is a single OPC client that reads the state of the virtual worlds of all teams and stores the data in a database.

Scoring System The *scoring system* reads the data in the virtual world monitor's database and determines the team scores for the CPS component.

Virtual World Display The *virtual world display* is used to display virtual world data and/or team scores.

The virtual world display can be either a video monitor, a physical representation of the virtual world, or a combination of the two. The use of a display on the backend, instead of as part of the virtual world, allows the same display to be shared by multiple teams.

4 CPS-CDC: Case Study

Recently, we hosted a CPS-CDC using our design from the previous section. This section describes the specific scenario we developed for the competition, including details about the CPS devices and the virtual world.

4.1 Cyber-Physical Scenario

Our CPS extension supports essentially all types of CPS devices, so the possible CPS-CDC scenarios are many. As a virtual world, we choose to simulate a city. For the physical services simulated, we choose the city's power and water grids, since modern life depends on these services. The power grid is heavily cited in the news as an example of the cyber-dangers facing our nation's infrastructure. In addition, power can be intuitively represented in the physical world through the use of LEDs, providing a nice virtual world display mechanism. We choose to also simulate water because it is another critical element of a city's infrastructure.

To create a virtual CPS service for a scenario, three of the four main components discussed in Section 3 must be customized to fit the service. A service needs CPS devices, an HMI for the blue teams to interact with their CPS devices, and a virtual world implementation that defines how the devices interact with each other and the virtual world. The following sections describe how we customize these components for the water and power services of our virtual city.

4.2 CPS Service: Water

4.2.1 Water CPS Devices

As shown in Fig. 4, our virtual water system, which divides the city into quadrants, is comprised of six pumping stations and a water treatment center. The pumping stations are dependent on one another according to the flow shown in the figure; for example, bringing down pumping station 4 only affects that station (and thus water for that quadrant of the city), whereas bringing down pumping station 1 also brings down stations 2–5 (and shuts off water for all quadrants of the city).

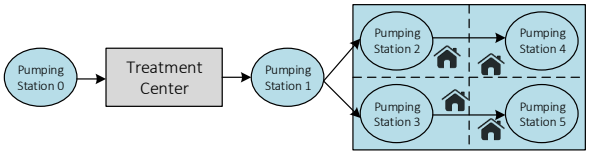

Figure 4: Overview of the water treatment network.

Each node shown in Fig. 4 is composed of multiple devices, with each device represented by one tag. Each pumping station has two pumps, two valves, and a chlorine tank. The treatment center is more complex, composed of nine automated valves, three pumps, seven chemical tanks, an ozone contractor, and three UV chambers.

The pumps and valves affect the flow of water. The remaining components (*sanitation units*) affect the water quality and are modeled after the water treatment process of the Winnipeg Water Treatment facilities [16].

Some of the devices in both the pumping stations and the treatment center are redundant, increasing the challenge for the attackers, who generally do not have access to the HMI or know the topology of the process represented by the tags. Attackers may simply try to write to all tags in order to cause more disruption, but this strategy is also more likely to alert the blue team to the intrusion before a disruption occurs.

In the CPS-CDC, all the modifiable tags in the water network are Boolean values representing whether the device is operational or not. An additional tag (with a floating point value) represents the water quality, and is automatically calculated as the percentage of water sanitation units that are operational at the current time.

4.2.2 Water HMI

The water HMI serves as the front end to the virtual water service. The HMI is intended to be as simple as possible, while still capturing the core concepts of SCADA for the water service. The HMI application is designed with the following requirements:

Control The application must provide an interface allowing operators to manipulate the state of tags for the connected process.

Error Reporting The application should promptly notify the operator of errors in the network.

Logging The application should provide logging functionality so that operators can review recent events.

Simplicity The application should be intuitive to use. Furthermore, operators should be able to accurately infer the state of the process though the display.

Industrial-style User Interface The application should mimic the look and feel of an authentic SCADA software, in order to provide an extra level of depth to the CPS-CDC environment.

When users first launch the water HMI, they are presented with an overview panel of the water treatment network. From there, they can control the water service and monitor for any anomalous behavior. The HMI is implemented using wxPython for the GUI and PyOPC for the communication with the CPS devices. Fig. 5 shows a screen-shot of one of the interactive HMI panels.

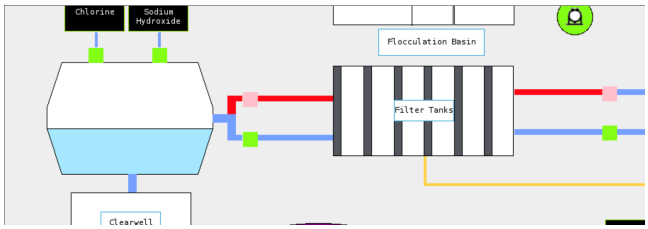

Figure 5: Example of one of the interactive HMI panels.

4.2.3 Water Virtual World Implementation

The set of rules that govern the virtual water service is implemented on the virtual world host of each team. As previously discussed, the failure of some nodes in the water service network also causes the failure of dependent nodes. This cascading failure effect should be gradual, simulating the time it would take for these systems to influence one another in the real world.

To achieve this, a *service scanner script* runs on the virtual world host of each team and constantly monitors the status of the tags in the water network. The script uses custom service scanner objects, where each object monitors its own pumping station or treatment center in the overall network. Each scanner object has a linked list of objects that represent the tags in the process being scanned. These tag objects together form a graph of the water flow in the pumping station or treatment center. Each tag object stores the tag's name, its fail value, the comparison operator for the fail value, its previous neighbors, and the next node or nodes in its path.

The scanner script runs at a periodic interval. It first polls for updates to the tags it is monitoring. When it finds an update, it stores the new value in its corresponding tag object. If the value causes the device to change to a failure state, the object notifies its forward neighbors. Each forward neighbor then enumerates its upstream nodes to determine whether there is at least one active path from the source to itself. If it finds such a path, it remains in its current state; otherwise, it: (1) updates its own internal state to "disabled," (2) updates its state in the virtual world, and (3) notifies its own forward neighbors. This process continues until the end of the graph is reached. At the end, there is a special tag that is invisible from the competition network. If this tag is enabled (receiving water flow), it means this segment of the water service network is operational; otherwise, it signals to the other segments of the water service network that it is down.

To communicate changes in the virtual world (e.g. cascading failures) back to the CPS devices, the OPC server in the virtual world host is continually polled by the VCS layer of the devices. Any updates will then eventually propagate to the HMI, which polls the VCS layer of the devices.

An interesting property of the water treatment service scanner script is that it implicitly enforces a specific start sequence for the water treatment network. The network cannot be brought up in any arbitrary order, because for a node to become active, there must be at least one active path that reaches the node. Any attempt to start devices where there is no active flow will result in those devices being disabled after a pass from the service scanner. Note that sanitation units, while requiring an active flow to be enabled, do not themselves influence the flow when disabled, but instead weaken the status of the overall water quality.

4.3 CPS Service: Power

4.3.1 Power CPS Devices

We model the power grid for the virtual city in two halves, each with a generator and a load. Each half has four tags: the generated power value, the generator breaker value, the load's power value, and the load's breaker value. One additional, invisible tag is used to indicate the status of the virtual power system. When either half of the power grid loses power, this tag is disabled to signify a power outage.

4.3.2 Power HMI

The power HMI is divided into two interfaces that depict the two halves of the power system. Each user interface has three gauges representing generation, load, and power flow. Input fields allow the operators to increase or decrease the power generation supplied by the generator. In addition to the generator input fields, there are two breaker buttons that correspond to the breaker attached to the generator and the breaker attached to the load. Operators can trigger either of these breakers to remotely disable the flow of power to that portion of the grid.

4.3.3 Power Virtual World Implementation

Similar to the virtual water service, the virtual power service has a service scanner script that checks the values of the breakers, as well as the current load on the network, to determine if power is flowing or not. The script first verifies whether the generator breaker or load breaker has been tripped. If so, power has been disrupted for that half of the power grid.

Otherwise, the script compares the values of the generator power and the load power. If the generator power is less than the load power, power has been disrupted for that half of the power grid. Disruption to power has an instantaneous effect on the network. If the scanner finds power has been disrupted to either half of the city for any reason, it sets the invisible status tag to zero.

4.4 Virtual City Display and Anomalies

A major goal of a CPS-CDC is to incorporate physical elements into the testbed and competition. Physical elements include the use of actual control systems, embedded systems or equipment, and situations that are acted out in the physical world rather than a virtualized environment. In our CPS-CDC, we accomplished this goal through the use of physical displays for our virtual city, and through the use of anomalies.

Our two physical displays are a physical model of the city, shown in Fig. 6, and a demonstration water pump. The physical model of the city was designed by Prof. Leslie Forehand of College of Design at Iowa State University, and her students. The demonstration pump consists of a water pump attached to a microcomputer and a miniature water tower. We connect this system to the HMI of a demonstration blue network in order to show competition participants how the virtual world interacts with the physical hardware.

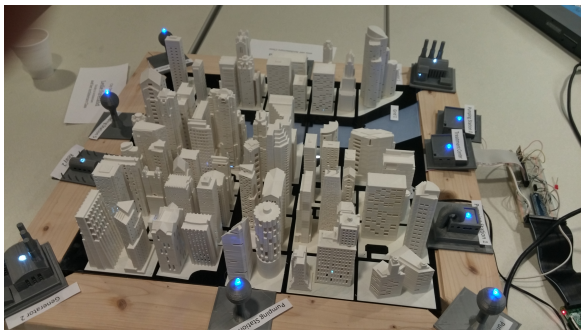

Figure 6: The 3D-printed city that represents the state of the world for each team.

Our physical model of the city is 3D-printed, with each utility given its own building that represents its services. We wire the 3D city with color-coded LEDs to indicate the state of the services. Blue LEDs represent water services and white LEDs represent power. The LEDs are connected to a Raspberry Pi microcomputer that queries the virtual world aggregator about the state of the world for each team. During the competition, the city cycles through each team, displaying the current state of services in that team's virtual world.

During the competition, we generate anomalies that ask teams to send team members to the 3D city to visually inspect the state of their services. The purpose of this exercise is to simulate utility employees making on-site calls. For example, employees of an electric utility may have to travel to find the source of disruption to the power grid, such as downed power lines or blown transformers.

Additionally, depending on the depth of Red Teams penetration in the CPS network it may be required for Blue team members to physically inspect the physical 3D model to ensure that the HMI is accurately reporting what is occurring in the environment. This provides a way for Blue team to take defensive actions in case Red team spoofs network traffic.

The 3D city also helps to make the CPS-CDC a better spectator sport. In a traditional CDC, spectators generally only have the scoring system to monitor and use to infer the state of events in the competition. Using the 3D model of the city and other display mediums, we can enhance the experience for spectators by creating exciting, visible representations of real-time events.

5 Observations and Experiences

In this section, we summarize a few observations and experiences from the competition.

Overall, the addition of the CPS component was very beneficial in reinforcing skills such as network segmentation to the competitors. Blue teams got a chance to learn about CPS systems, SCADA, HMIs, and OPC. They needed to understand how the CPS systems work in order to defend against attacks. The red team was able to successfully shut down different portions of the water system and power grid for most teams. However, as red team members mainly interacted with the CPS devices, blue teams were able to quickly notice, via the HMI, when services were taken down. A CPS-CDC would provide a more realistic experience to participants if an attacking tool is available to the red team that can adjust the HMI to produce erroneous observations about what is happening on the CPS devices. This is part of our future work. Another lesson learned from this project is that we may want to introduce more anomalies to the CPS systems to make the competition more realistic. In our CDC, blue teams were able to spend most of their time in network defense, while only casually checking on the CPS components to make sure everything was running correctly.

To simplify the deployment of the CPS-CDC, we focused on services delivered to large partitions of the city. This included two power grids and four quadrants for the water supply. For future competitions, the service areas could be further refined to include individuals blocks and even homes. This would allow us to represent a wider variety of failures, such as damage to power lines or water issues caused by flooding. This could allow us to use natural disasters as part of anomalies. When coordinated with attacks from the red team, such an anomaly would put considerable pressure on blue teams, as well as expand the range of their responses.

We plan to release the developed CPS component, together with the Internet event simulator, to the cyber security re-

search and education community. The package will include (1) a detailed specification of the required hardware to set up the testbed; (2) a downloadable software package that includes our Internet event simulator and the CPS component, with detailed documentation, a user manual, and installation instructions; and (3) a tutorial on how to set up the CPS-CDC.

6 Conclusion

In this paper, we describe the design and implementation of a cyber-physical system (CPS) extension to complement our Internet event simulator at Iowa State University, known as ISEAGE, which has been used for cyber defense competitions (CDCs). In the future, we plan to introduce other types of CPS into ISEAGE and CDC, such as ITS (Intelligent Transportation Systems) and IoT (Internet of Things). Also, we plan to work on a generic solution for creating the HMI interface and the state machines and scanner scripts of the virtual world. While the implementation proposed in our current CPS-CDC is quite scalable, steps could be taken to make the introduction of new CPS scenarios easier.

Acknowledgement

This work is funded in part by the U.S. National Science Foundation under Grant No. 1730275. We thank Dr. Julie Rursch for her valuable advice and suggestions to this work.

References

[1] K. Zetter, "Inside the cunning, unprecedented hack of ukraine's power grid," *Wired*, 2016.

[2] J. A. Rursch and D. Jacobson, "When a testbed does more than testing: The internet-scale event attack and generation environment (iseage)-providing learning and synthesizing experiences for cyber security students." in *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2013, pp. 1267–1272.

[3] J. Lepreau. Emulab.net: An emulation testbed for networks and distrubed systems. [Online]. Available: www.cs.utah.edu/flux/testbed-docs/testbed-intel-jun01.ppt

[4] T. Benzel, "The science of cyber security experimentation: The deter project," in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011.

[5] E. Jaffe and J. Albrecht, "Planetlab-p2p testing in the wild," in *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, 2009, pp. 83–84.

[6] E. A. Lee, "Cps foundations," in *Design Automation Conference*. IEEE, 2010, pp. 737–742.

[7] C. W. Axelrod, "Managing the risks of cyber-physical systems," in *Systems, applications and technology conference (LISAT), 2013 IEEE Long Island*, vol. 6, 2013, pp. 3–3.

[8] Y. Peng, T. Lu, J. Liu, Y. Gao, X. Guo, and F. Xie, "Cyber-physical system risk assessment," in *2013 Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2013, pp. 442–447.

[9] W. Yang and Q. Zhao, "Cyber security issues of critical components for industrial control system," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. IEEE, 2014, pp. 2698–2703.

[10] "Industrial control systems." [Online]. Available: https://www.shodan.io/explore/category/industrial-control-systems

[11] C. Wang, L. Fang, and Y. Dai, "A simulation environment for scada security analysis and assessment," in *2010 International Conference on Measuring Technology and Mechatronics Automation*, vol. 1. IEEE, 2010, pp. 342–347.

[12] Siemens, "Data exchange via opc xml." [Online]. Available: https://cache.industry.siemens.com/dl/files/938/27097938/att_78439/v1/faq_opc_xml_da_datenaustausch_v10_en.pdf

[13] H. Himmelbaur, "Pyopc." [Online]. Available: https://sourceforge.net/projects/pyopc/

[14] FreeOpcUa, "python-opcua." [Online]. Available: https://github.com/FreeOpcUa/python-opcua

[15] "Build root." [Online]. Available: https://buildroot.org/download.html

[16] "Drinking water treatment plant." [Online]. Available: https://www.youtube.com/watch?v=20VvpASC2sU