Evaluating the Effects of Cyber-Attacks on Cyber Physical Systems using a Hardware-in-the-Loop Simulation Testbed

Bradley Potteiger, William Emfinger,
Himanshu Neema, Xenofon Koutosukos
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN 37235

CheeYee Tang, Keith Stouffer
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

Cyber-Physical Systems (CPS) consist of embedded computers with sensing and actuation capability, and are integrated into and tightly coupled with a physical system. Because the physical and cyber components of the system are tightly coupled, cyber-security is important for ensuring the system functions properly and safely. However, the effects of a cyber-attack on the whole system may be difficult to determine, analyze, and therefore detect and mitigate. This work presents a model based software development framework integrated with a hardware-in-the-loop (HIL) testbed for rapidly deploying CPS attack experiments. The framework provides the ability to emulate low level attacks and obtain platform specific performance measurements that are difficult to obtain in a traditional simulation environment. The framework improves the cybersecurity design process which can become more informed and customized to the production environment of a CPS. The developed framework is illustrated with a case study of a railway transportation system.

Keywords

Hardware-in-the-Loop, Model Integrated Computing, Vulnerability Assessment, Resilience, Cyber-Physical System, Experimentation, Testbed

I. Introduction

Cyber-Physical Systems (CPS) include co-engineered interacting networks of physical and computational components. Such systems typically consist of embedded computers with sensing and actuation capability, and are integrated into a tightly coupled physical system. Because the physical and cyber aspects of the system are tightly coupled, cyber-security is important for ensuring the system functions properly and safely. Cyber-security is further becoming increasingly important as many CPS are becoming distributed and utilizing wired or wireless networks for communications and coordination. Such distribution enables smarter systems with increased functionality but also creates a larger attack surface.

In traditional cyber-security it can be difficult to determine how a cyber-attack will affect the running system, especially given that the same attack will most likely have different effects depending on which subset of the system is being attacked. Given how networks and communications couple the components of a system, determining how the effects of an attack propagate through the system can compound the difficulty of such analysis and of predicting attack severity. These problems are only exacerbated when such networked systems are connected to sensors and actuators which tightly couple the system to the physical world. In that case the attack's effects propagate not only through the cyber and communications parts of the system, but also through the embedded controllers and into the physical world [12] [5].

Our previous work focused on utilizing the Command and Control Wind Tunnel (C2WT) environment to develop synchronized, multi-domain CPS simulations [1]. By combining different CPS models (Network, Physics, User Interface) into a single system of systems simulation, a vulnerability assessment framework was developed to analyze impacts of cyber-attacks on different levels of a CPS. Further, an attacker-defender (Red team vs Blue team) game was implemented to aid in developing cybersecurity strategies. However, in certain cases such as platform dependent vulnerabilities, simulations are limited in their ability to predict system behavior. This is especially true for attacks such as distributed denial of service (DDOS) and code injection attacks that have platform dependent impacts.

To address the difficulties of performing impact analysis on cyber-attacks in CPS solely on simulation techniques, we have extended a hardware-in-the-loop (HIL) CPS testbed, and an associated experimentation software development and deployment platform. The HIL testbed extends the capabilities of the C2WT to emulate the CPS software on embedded computers that are representative of the future production environment.

The work at Vanderbilt is supported by NIST (70NANB13H169).

Realistic measurements of the system behavior in the presence of cyber-attacks and defense mechanisms can be obtained, which in turn can be fed back into the simulation environment to provide more accurate results. The testbed and software platform together allow rapid development and experimentation with a variety of CPS, ranging from networked satellites, to airplanes and cars [8] [9]. In this paper we demonstrate how the model based software development framework can be used with the testbed for rapidly deploying cyber-attack experiments for impact analysis in a safety-critical CPS such as a railway transportation network. Railway presents a critical domain in that hazardous material, shipping companies, and the general public rely on this method of transportation for timely, reliable and safe movement.

The paper is organized as follows: Section II discusses our previous research and related work, Section III details the architecture of the HIL testbed, Section IV presents a model driven experiment development software framework, Section V presents a case study based of a railway transportation system illustrating the testbed and software framework, and Section VI provides concluding remarks for the paper.

II. Related Work

Cyber-security for CPS is a rapidly growing field, as researchers are demonstrating critical vulnerabilities in networked CPS such as automobiles [2], implanted pacemakers [4], [6], and home automation systems [12]. However, testing the security of networked CPS is challenging, both with regard to the difficulty and price of setting up a CPS on which to test and with regard to the possible dangers associated with the results of the experiments [12] [13] [14]. This is especially true for larger-scale safety critical CPS, such as transportation networks, power distribution networks, and commercial airlines. For such systems, a real laboratory testbed is impossible, leading to the development of simulation or emulation testbeds which involve a simulation of the CPS. The authors in [12] demonstrate the need for complete testing of CPS, as even non-networked devices such as standard lightbulbs can be compromised and pose a threat to users if a networked home automation controller becomes compromised. Such a threat pathway clearly indicates that the physical characteristics of the CPS and its environment play a critical role in threat analysis.

Since this paper presents the uses of such a testbed and how it can be used to analyze the behavior of a distributed CPS under attack, we will not cover the advances in many of the related fields associated with simulation of physical systems, hardware in the loop simulation, software platforms for experiment development, deployment and analysis, or cyber-attack detection and mitigation. This paper addresses only the goal of analyzing how cyber-attacks propagate through networked CPS through the software and into the physical domain. More details about the design of the testbed, its integration with physics simulation, and the software platform enabling rapid experimentation on the testbed can be found in [8] and [9], respectively.

Security research in CPS using testbeds has been an active area of research. A good overview of their architecture, and their application in the Smart Grid domain is provided in [5]. The authors mention nine research applications of security research using CPS testbeds as: 1) vulnerability research, 2) impact analysis, 3) mitigation research, 4) cyber-physical metrics, 5) data and model development, 6) security validation, 7) interoperability, 8) cyber forensics, and 9) operator training. Many of these applications are related, but our main research application according to this classification scheme is #2, viz. impact analysis, which "explores the physical system impacts from various cyber attacks to quantify physical system impact."

Indeed other CPS testbeds have been built providing varying degrees of fidelity with respect to the CPS they are trying to analyze. Supervisory Control and Data Acquisition (SCADA) systems have been explored most extensively [10] [3] since they form the backbone of much of the critical infrastructure.

Many of these alternative platforms for security testing do not incorporate the CPS (either through simulation or in the loop), are tied to a specific system being tested, and/or are prohibitively expensive to replicate or extend in other research labs. The rest of the paper demonstrates how the HIL CPS testbed can provide similar capabilities for impact analysis without being bound to any specific CPS.

III. HIL Architecture

To determine, measure, and analyze the effects of cyber-attacks on networked CPS, we need a platform for developing tests and experiments which can detect, quantify, and measure the effects of cyber-attacks on real CPS. Such a platform needs to have real embedded hardware that would be used in the CPS, and this hardware must have a way to sense and actuate a physical system. For many systems, building and deploying the real CPS is not feasible or not possible due to financial, logistical, or safety reasons. It is traditional and acceptable to use physics simulators to act as the CPS, using hardware-in-the-loop (HIL) with simulation to provide the computational and communication capabilities of the CPS. Additionally, a complete software development and deployment infrastructure is required to enable rapid, iterative experiment design, deployment, and data collection.

The rest of the paper covers the procedures for developing and deploying security experiments using our model based software development framework. To complete the description of the experimental environment, we first describe the architecture of both the hardware and software platforms used by the HIL testbed.

A. Hardware Architecture

The functions of a CPS testbed for security research are 1) (re-)configurability with respect to CPS and software, 2) accurate behavior of the software with respect to the CPS, 3) accurate behavior of the network with respect to the CPS, and 4) accurate behavior of the sensors/actuators with respect to the CPS. In addition to these concerns, the CPS testbed should behave similarly to the real system in the case of failures or attacks.

Because of these requirements, the CPS testbed was extended with a Hardware-in-the-Loop Emulation platform, where embedded computing nodes on multiple configurable networks are connected to one or more simulation machines. The simulation machines provide the embedded computers the ability to sense and control the (simulated) physical systems in which they will be deployed. The configurable network through which the embedded computers communicate allows more robust and higher-fidelity emulation of the CPS' network as it would be deployed than if the network were simulated. An architectural diagram of the CPS testbed is shown in Figure 1. For more information about this testbed please see [8].

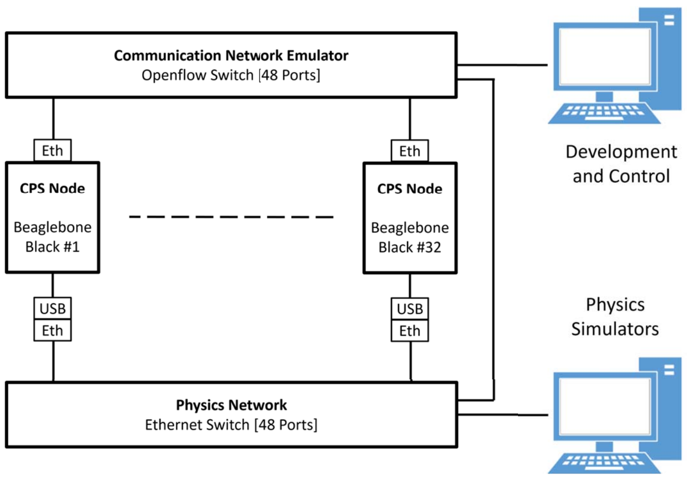

Fig. 1: Diagram showing the architecture of the CPS security HIL testbed.

B. Software Platform

In addition to the hardware and simulation platform which form the backbone of the HIL CPS testbed, the software platform must provide functions for 1) developing the CPS code, 2) developing the attack and measurement code, 3) configuring the experiment, 4) deploying the experiment, and 5) retrieving the results from the experiment for analysis. To meet these requirements, the ROSMOD software toolsuite is extended [9]. Using this software platform, we can create reusable software components which each provide key functionality for our experiments. We compose these software components together into an experiment for deployment onto the HIL CPS testbed. Further details about the software framework are detailed in the following section.

IV. Model Based Software Development for CPS Security Analysis

For the purpose of vulnerability analysis on the HIL testbed, a software framework is used to enhance and streamline the cyber-attack experiment creation process. ROSMOD is a graphical model integrated computing(MIC) tool that utilizes component based design principles with the Robot Operating System (ROS) middleware for representing the distributed nature of a CPS. The ROSMOD framework is extended to support the implementation of cyber-attacks within the component and communication scope for the purpose of exploiting respective vulnerabilities in a component's software. The rest of the section details the process of creating an experiment and the different capabilities that can be included.

A. Experiment Development

In respect to the hardware-in-the-loop testbed, the purpose of the software framework is to enhance and allow for rapidly prototyping experiments for measuring the effects of cyber-attacks on a CPS. For accomplishing this task, experiments can be developed representing a system at baseline, as well as under attack to identify potential vulnerabilities and analyze the resiliency and fault tolerance of critical infrastructure. ROSMOD provides the ability to model the specific components of an experiment, as well as the communication network through the ROS protocol. ROS utilizes a publish-subscriber technique where each line of communication is assigned a unique identifier (ROS Topic) which can be used by subscriber nodes to filter out incoming messages. To successfully develop an experiment in the software framework, the following needs to be completed:

1) Development of the Software Model
2) Development of the System Model
3) Development of the Deployment Model
4) Execution of the Experiment Interpreter

The first step involves developing the generic component libraries representing the various parts of a CPS. The component in the software library represents a basic building block for insertion into an experiment including timer based code execution, publisher-subscriber functionality for communication, and variable placeholders for allowing components to be customized after being placed into an experiment model. Furthermore, the communication messages and their contents are defined, as well as external libraries to link to simulator application program interfaces(APIs). A compilation interpreter is included to provide the ability to cross compile the library code under the testbed node architecture(Arm V7 architecture) and identify locations of errors within the code in the graphical interface similar to a traditional compiler.

The next step includes defining the architecture of the hosts that the experiment is executed on. This includes defining attributes such as the host name, operating system architecture, and location of a secure shell(ssh) key for accessing the host. Additionally, the model specifies the connection of the host to the ROSMOD server through an internet protocol(IP) address for the purpose of transferring and executing code on the hosts and fetching the results back to the ROSMOD server.

The third step includes taking the generic component building blocks previously built in the software library, and customizing them to produce an implementation model for the experiment. This includes creating component instances and editing the component parameters appropriately through the variable placeholders. The deployment represents the finalized experiment model to be executed on the testbed.

The final step involves executing the experiment interpreter which maps the deployment components to the appropriate host nodes declared in the system model (HIL embedded computers), transfers each respective component binary to the appropriate host, and starts the component processes on the HIL testbed nodes. Additionally, this is the time where the user starts up the physics simulator for synchronization with the respective component processes. Furthermore, after the simulation is complete, the results are fetched from the respective hosts and transferred to the server where they can be analyzed in the ROSMOD graphical environment.

B. Experiment Capabilities

The following features can be included in experiments using the development process described above.

1) Baseline: For the purpose of measuring the effects of attacks as well as the impact of defense mechanisms, the first place to start is with a baseline experiment under normal operating circumstances. In this case, the CPS is implemented as it would operate with no external impacting factors, as in how the operators expect the system to behave on a routine day to day basis. This allows the operators to have data to compare against when trying to decide whether an attack is occurring and have a data-centric method of determining the best course of action to mitigate the situation.

2) Attack Implementation: To effectively identify the most critical vulnerabilities of a system, it is important to test the system under as many circumstances as possible, including under attack. By developing experiments with individual attack implementations, the attack effects on system specific metrics are measured to appropriately quantify the impact of the attack and in turn the criticality of the system vulnerability. This aids the risk assessment process in allowing engineers to have an objective, quantitative metric to judge various vulnerabilities. Various attacks are implemented to effect the confidentiality, integrity, and availability of a CPS. Confidentiality attacks that are implemented include packet sniffing, password/authentication, session hijacking, and side channel attacks. Integrity attacks include replay, spoofing, packet delay, and packet dropping attacks. Availability attacks include denial of service, distributed denial of service, and spamming attacks. Additionally, attacks are implemented that include multiple categories such as buffer overflow and code injection attacks.

3) Attack Linking: To make attacks more realistic, the software framework provides the feature of staging multiple attacks to occur in an experiment. As such, multiple attacks can be linked together to identify propagating vulnerabilities in a system and maximize the impact of an attacker. All of the attacks described above can be combined into a single experiment implementation. Attacks can be modeled both occurring simultaneously and occurring in stages. For the simultaneous category, attacks that are independent can be implemented within the software model with no corresponding communication with one another. For example, with an experiment including a denial of service attack on a respective component and spoofing attack on a separate part of the network, the two attacks are not correlated to one another and can be implemented in software as occurring through the entire simulation with no communication or timer dependencies. However, it is often the case that attacks are dependent on one another. As such, attacks can be modeled as being dependent on timer information, event information, or both. For example, by accessing timer data in the case of the denial of service and packet spoofing attack, the packet spoofing attack can be deployed to start at the beginning of the simulation while the denial of service attack can be programmed to start at 200 seconds into the simulation. Additionally, with a case such as a replay attack and packet sniffing attack, the replay attack will be dependent on the success of the packet sniffing attack capturing a transmitted packet. As such, event based dependencies can be programmed into the software model by having communications between the respective attacks. For example, once the packet sniffing attack is successful, an event message can be transmitted to the replay attack implementation including the captured packet information. Once this message is received, the replay attack sequence can be initiated and executed.

4) Defense Mechnamisms: The most important benefit to system engineers is the ability to implement and analyze the effects of defense mechanisms on the security and functionality of the CPS. The insertion of a defense mechanism often produces a tradeoff (e.g., performance vs. security) where the benefit has to be weighed against the cost. By measuring the performance overhead of the defense mechanism on the testbed while analyzing the success of preventing the targeted attacks, engineers can make the crucial cost benefit decision when designing their systems. Engineers can implement defense mechanisms dealing with hardening communications (AES 256 message encryption, message authentication, firewall implementations, dynamically changing ip addresses), hardening the component device software (password authentciation, input buffer limits, variable scope enforcement), or implementing monitoring algorithms for detecting abnormalities (threshold detection, gaussian processes, unsafe state detection).

5) Data Collection: After the conclusion of an experiment simulation, it is important to analyze the results to determine the behavior of the system. As such, the software framework provides the ability to fetch data results to analyze through the graphical interface. During every experiment, event times are logged in the host repositories and transferred back to the ROSMOD server at the end. By default these events include the timer based execution of code, publishing of messages, and receipt of subscribed messages. Additionally, custom log messages can be inserted to record experiment-specific events such as when a state is changed in a controller. In the graphical interface, the data can be downloaded in text files, but events can also be observed through time domain plots, allowing for rapidly analyzing the sequence of events in the experiment. Furthermore, the user can generate data from the physics simulator to record the consequential behavior in the physical environment. This allows for analyzing the cyber effects of the experiment on the physical environment.

V. Case Study

The reference case study is based on a railway transportation system. In this example, there are many railway signals and switches that route trains throughout the rail network. Railway signals have a green or red state and determine whether a train can travel to the next rail segment. In circumstances where a junction exists that connects multiple rail segments, rail switches are used to route trains to the appropriate adjacent rail segment. Each rail switch or signal is controlled by command messages sent through a communications network by a train operator located at a central facility. The communication network is comprised of network switches, routers, and basestations. The communications from the train operator are first transmitted through ethernet to a network switch and then passed to a router which transmits packets wirelessely to basestations in the field. Basestations then filter out and relay command packets to the appropriate associated rail signals and switches at the location.

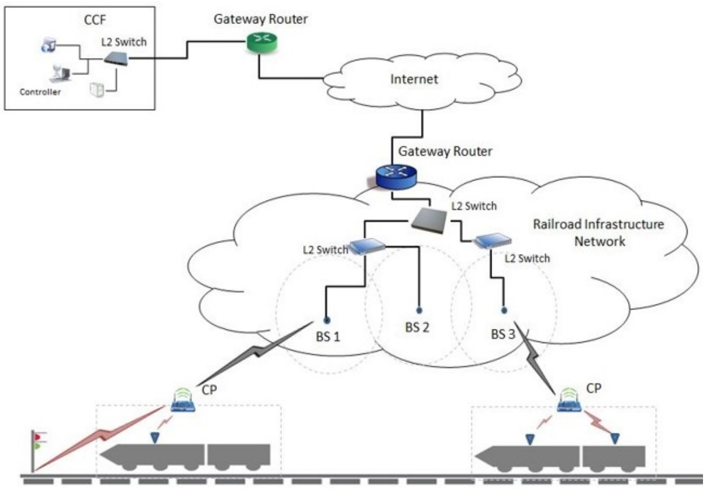

Fig. 2: The railway communication network

For the CPS cyber-attack experiment, a critical rail segment is selected for attack which serves as a central hub to the rest of the network. The corresponding rail segment as well as its relationship in the communications network and connection to the railway network simulated in Train Director [7] are shown in Figure 3. This example consists of ten rail signals, six rail switches, four basestations, two routers, a network switch and a train operator. These components from the architecture described in Figure 2 are mapped to individual nodes in the HIL testbed for execution of the experiment.

A. *Experiment 1: Packet Delay Attack*

The method of attack on the railway network consists of a man in the middle attack on the communications to the Basestation 1 software component. The railway network is comprised of actuators (railway signals and switches), as well

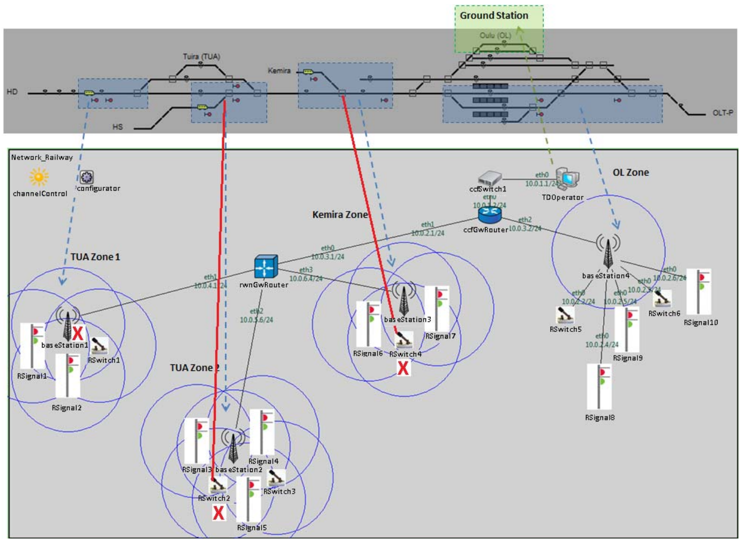

Fig. 3: Train Director Railway Network Configuration and Communication Network Relationship

as communication infrastructure (basestations, routers, network switches) that relay command messages from a train operator located at a central facility. The wired network between the train operator, network switch, and router is encrypted and authenticated. Additionally, the wireless communication between the routers and basestations has cryptographic encryption but no authentication, preventing an attacker from spoofing command packets. However, this doesn't prevent an attacker from inserting a malicious node to delay the routing of command messages through the network.

In this experiment, command packets (ROS Messages) from the train operator ROS node consist of an actuator name and goal state string variable. The attacker creates a malicious man in the middle node and intercepts traffic routed through Basestation 1. As such, the attacker has the ability to delay when command packets are transmitted to the Rail Signal 1, Rail Signal 2, and Rail Switch 1 components. The attacker uses this vulnerability to execute a packet delay attack, delaying command packets by 20 minutes to have a sufficient effect on the railway network to keep trains from reaching their destinations on time, but at the same time, decreasing suspicion of a denial of service attack or faulty components by keeping components operational. A diagram illustrating this experiment is shown in Figure 4.

1) Experiment Configuration: Configuration of the experiments entails:

1) Configuring the communications network between the software components,
2) Mapping the software components to simulated CPS sensors/actuators as required,
3) Configuring the parameters of the attack, e.g., amount of time to delay command messages,
4) Mapping the CPS software components onto the processing hardware of the testbed,
5) Mapping the attacking software components onto the

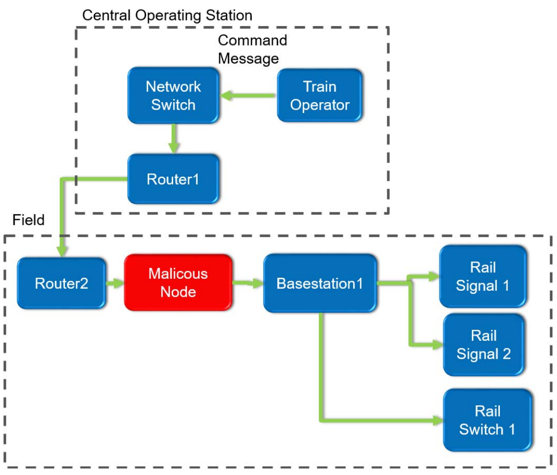

Fig. 4: The attack vector used against the railway communication network

processing hardware of the testbed, and

6) Configuring the simulator (Train Director) with the proper rail network and communications interface

The ROSMOD software infrastructure described previously enables all of this configuration from its graphical user interface(GUI), as well as deployment, startup, and shutdown of the experiment. For the experiment, the malicious man in the middle node delays packets for 20 minutes from the time the message was received. There are also rail signals, rail switches, basestations, routers, network switches, and a train operator component in the deployment. Each component is mapped to its own computing node.

Further, the experiment development process using the software framework includes developing a software model, developing a system model, developing a deployment model, and executing the experiment by running the experiment interpreter. To develop the software model, component specific code is written to serve as the basic building blocks for designing the CPS topology. Additionally, external libraries are linked into the framework for accessing simulator APIs. To develop the system model, the HIL testbed architecture is defined including the network connectivity, hostname, operating system, ssh key, and IP address for each node in the testbed. To develop the deployment model, instances of generic component models from the software library are customized to create the experiment implementation model. This includes designing the the CPS topology, as well as implementing cyber-attack instances. To run the experiment, an interpreter is executed to map the deployment model to associated nodes in the HIL testbed, transfer the compiled component binaries to the appropriate hosts, start component processes, run the simulation, and fetch results at the end of execution.

2) Results: From the experiment configuration explained above, we want to determine the effects that the packet delay attack (which is a cyber-attack) has on the train arrival times in the network. For the physical system, which is simulated in Train Director, we directly use Train Director to measure the effects of the attack. The attack alters the behavior of the rail signal 1, rail signal 2, and rail switch 1 components and we wish to determine how these changes affect the train delay in the railway network. We directly capture the state of the railway network from Train Director represented by the time the train arrived late to its destination.

Determining and measuring the effects of the packet delay attack on the software's behavior is not difficult, given that the behavior and structure of the software component are known. However, given the coupling between the software and the physical system, it is not as easy to determine what the effects of the altered behavior will be on the overall CPS as a whole. Clearly, the delayed reception of the train operator command messages will cause any trains at the affected location to be delayed. However, this alteration in behavior does not lend itself directly to prediction of the state of the train delay since it is difficult not only to quantify but to relate to the train routes. With the physical simulator (Train Director) we are able to see how the trains respond to the degraded behavior of signal and switch actuators and measure the effects on the train arrival times. Figure 5 shows the results of the Train Director simulation in respect to the lateness of the respective trains. As can be observed from the figure, the packet delay attack caused considerable delay to a number of the trains. Additionally, the trains that were not on the path were not affected by the attack. Therefore, if an attacker wanted to limit his exposure while maintaining the goal of delaying a critical shipment on this route, this attack could suffice. Furthermore, the delay of trains 4 and 5, which follow the path of train 2, is compounded by the time it takes train 2 to pass through the attacked signals and switches.

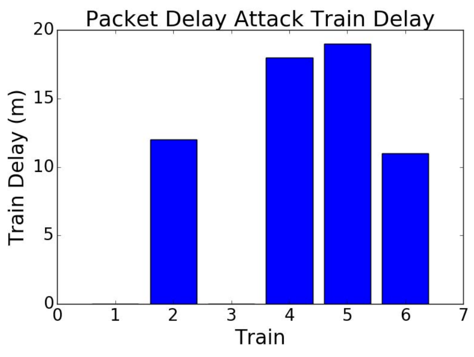

Fig. 5: Packet Delay Attack Experiment Train Delay

B. Experiment 2: Packet Delay Attack and Denial of Service

In addition to implementing a single attack in an experiment, the software framework provides the ability to create an experiment with multiple attacks. As such, the first experiment in which a packet delay attack is implemented is extended to include a second attack: a denial of service attack on a set of railway switches. In this instance the goal of the attacker is to not only delay trains but to prevent the trains not on a specific path (trains coming from railway signal 1) from being able to reach their destination. As such, the critical shipments on these trains will not reach their customers causing a detrimental effect on the receiving companies operating status. To execute this attack series, the packet delay attack will consist of a malicious man in the middle node effecting the rail signal 1, rail signal 2, and rail switch 1 components while the denial of service attack will focus on compromising the rail switch 2 and rail switch 4 components. Due to the fact that rail switch 2 and rail switch 4 are no longer functional, trains originating from the top and bottom of the map will be stuck and will not be able to travel to their adjacent rail segments.

1) Experiment Configuration: The experimental setup is consistent with the same process followed in Experiment 1. As such, the same software component library and system model are used. However, when developing the deployment model, the user configuration section of the rail switch 2 and rail switch 4 components are customized to represent a denial of service attack in addition to the changes already made to the basestation 1 communications path from the previous experiment. In this experiment the compromised components are a malicious man in the middle node, the rail switch 2 component, and rail switch 4 component. This deployment setup is then transferred to the HIL Testbed hosts for execution through the experiment interpreter.

2) Results: The Experiment 2 results are similar to Experiment 1 in that a set of trains are significantly delayed due to the attacks. In the packet delay attack example, train 1 and 3 arrived on time while the rest of the trains arrived to their destinations late. In Experiment 2, trains 1 and 6 arrived on time while the rest of the trains appear significantly delayed. This can be observed from Figure 6. However, it is important to note that even though there is significant train delay in the plot, this delay is actually unbounded. This experiment was run under a 200 minute experiment time, and therefore the delay is limited to that restriction. In reality, due to the fact that trains 2, 3,4, and 5 all have to travel through paths passing through rail switch 2 or rail switch 4, these trains will never reach their destinations due to the denial of service attacks on the respective rail switches. Therefore, as the execution time for this experiment is increased, the output train delays will subsequently increase.

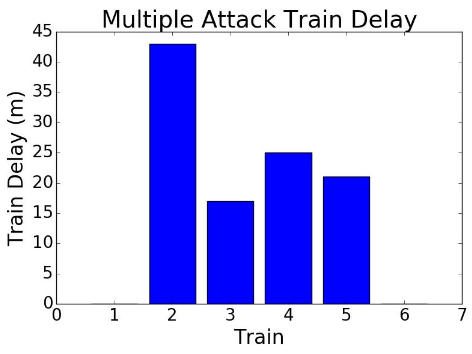

Fig. 6: Multiple Attack Experiment Train Delay

VI. Conclusions

In this work we've shown how a HIL CPS testbed can be used in security research for determining, measuring, and analyzing how cyber-attacks affect CPS systems and how the attack propagates through the software into the physical system. We described the architecture of such a testbed, including its hardware and software platforms and developed an example security experiment for a railway network. By measuring both the software behavior and the physical system behavior during both normal operations and under attack, we showed how the coupling between software and the CPS can be measured. This work is part of the first steps towards modeling, analyzing, and predicting CPS behavior during cyber-attacks, and by providing a testbed for systematically running experiments and collecting data for attacks, models and analysis techniques can be developed.

References

[1] Koutsoukos, X., Neema, H., Martins, G., Bhatia, S., Janos, S., Stouffer, K., Tang C.Y., Candell, R. Performance Evaluation of Secure Industrial Control System Design: A Railway Control System Case Study. In *4th International Symposium on Resilient Cyber Systems* (2016), Chicago.

[2] Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium* (2011), San Francisco.

[3] Davis, C., Tate, J., Okhravi, H., Grier, C., Overbye, T., and Nicol, D. Scada cyber security testbed development. In *Proceedings of the 38th North American power symposium (NAPS 2006)* (2006), pp. 483–488.

[4] Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., and Fu, K. They can hear your heartbeats: non-invasive security for implantable medical devices. *ACM SIGCOMM Computer Communication Review 41*, 4 (2011), 2–13.

[5] Hahn, A., Ashok, A., Sridhar, S., and Govindarasu, M. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *IEEE Transactions on Smart Grid*, 4.2(2013):847–855.

[6] Halperin, D., Heydt-Benjamin, T. S., Ransford, B., Clark, S. S., Defend, B., Morgan, W., Fu, K., Kohno, T., and Maisel, W. H. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 129–142.

[7] Train Director Raiload Simulation. *http://www.backerstreet.com/traindir/en/trdireng.php* Accessed on April 07, 2017

[8] Kumar, P., Emfinger, W., and Karsai, G. Testbed to simulate and analyze resilient cyber-physical systems. In *Rapid System Prototyping, 2015. RSP '15.* (October 2015).

[9] Kumar, P., Emfinger, W., Kulkarni, A., Karsai, G., Watkins, D., Gasser, B., and Anilkumar, A. ROSMOD: a toolsuite for modeling, generating, deploying, and managing distributed real-time component-based software using ROS In *Electronics 5.3(2016):53.*

[10] Mallouhi, M., Al-Nashif, Y., Cox, D., Chadaga, T., and Hariri, S. A testbed for analyzing security of scada control systems (tasscs). In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES* (2011), IEEE, pp. 1–7.

[11] Mitropoulos, D., Karakoidas, V., Louridas, P., Gousios, G., and Spinellis, D. Dismal code: Studying the evolution of security bugs. In *Proceedings of the LASER Workshop* (2013), pp. 37–48.

[12] Oluwafemi, T., Kohno, T., Gupta, S., and Patel, S. Experimental security analyses of non-networked compact fluorescent lamps: A case study of home automation security. In *Proceedings of the LASER 2013 (LASER 2013)* (Arlington, VA, 2013), USENIX, pp. 13–24.

[13] Van Leeuwen, B., Urias, V., Eldridge, J., Villamarin, C., and Olsberg, R. Cyber security analysis testbed: Combining real, emulation, and simulation. In *Security Technology (ICCST), 2010 IEEE International Carnahan Conference on* (2010), IEEE, pp. 121–126.

[14] Van Leeuwen, B., Urias, V., Eldridge, J., Villamarin, C., and Olsberg, R. Performing cyber security analysis using a live, virtual, and constructive (lvc) testbed. In *Military Communications Conference, 2010-MILCOM 2010* (2010), IEEE, pp. 1806–1811.