Integrated Simulation Testbed for Security and Resilience of CPS

Himanshu Neema
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
himanshu.neema@vanderbilt.edu

Bradley Potteiger
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
bradley.d.potteiger@vanderbilt.edu

Xenofon Koutsoukos
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
xenofon.koutsoukos@vanderbilt.edu

Gabor Karsai
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
gabor.karsai@vanderbilt.edu

Peter Volgyesi
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
peter.volgyesi@vanderbilt.edu

Janos Sztipanovits
Vanderbilt University
1025 16th Ave S, Nashville, TN
USA
janos.sztipanovits@vanderbilt.edu

ABSTRACT

Owing[1] to an immense growth of internet-connected and learning-enabled cyber-physical systems (CPSs) [1], several new types of attack vectors have emerged. Analyzing security and resilience of these complex CPSs is difficult as it requires evaluating many subsystems and factors in an integrated manner. Integrated simulation of physical systems and communication network can provide an underlying framework for creating a reusable and configurable testbed for such analyses. Using a model-based integration approach and the IEEE High-Level Architecture (HLA) [2] based distributed simulation software; we have created a testbed for integrated evaluation of large-scale CPS systems. Our tested supports web-based collaborative metamodeling and modeling of CPS system and experiments and a cloud computing environment for executing integrated networked co-simulations. A modular and extensible cyber-attack library enables validating the CPS under a variety of configurable cyber-attacks, such as DDoS and integrity attacks. Hardware-in-the-loop simulation is also supported along with several hardware attacks. Further, a scenario modeling language allows modeling of alternative paths (Courses of Actions) that enables validating CPS under different what-if scenarios as well as conducting cyber-gaming experiments. These capabilities make our testbed well suited for analyzing security and resilience of CPS. In addition, the web-based modeling and cloud-hosted execution infrastructure enables one to exercise the entire testbed using simply a web-browser, with integrated live experimental results display.

SAC 2018, April 9–13, 2018, Pau, France

ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/10.1145/3167132.3167173

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; *Dependable and fault-tolerant systems and networks* • **Security and privacy → Systems security** • **Computing methodologies → Modeling and simulation**

KEYWORDS

Cyber-Physical Systems, modeling and simulation, High-Level Architecture, security and resilience, courses of action

ACM Reference format:

Himanshu Neema, Bradley Potteiger, Xenofon Koutsoukos, Gabor Karsai, Peter Volgyesi, and Janos Sztipanovits. 2018. Integrated Simulation for Security and Resilience of CPS. In *Proceedings of ACM SAC Conference, Pau, France, April 9-13, 2018 (SAC'18),* 7 pages. DOI: https://doi.org/10.1145/3167132.3167173

1 INTRODUCTION

The last decade has seen an immense growth of internet-connected systems such as the Internet of Things (IOT) and several large-scale and learning-enabled Cyber-Physical Systems (CPSs) [1] [3]. CPS systems are often part of critical infrastructure such as railway, road transportation networks, power and water distribution systems, which makes them particularly attractive for adversarial attacks. In general, the attack vectors on these systems include attacks on the *physical infrastructure*, *computation hardware*, and *communication networks*. The tightly integrated nature of the physical, computational, and communication components enables attackers to use attack vectors in physical domain to attack the cyber domain and vice versa. Adding to this complexity is the fact that they usually span a variety of physical domains such as electrical, thermal, mechanical, and cyber (i.e. to route network packets for the sensor messages and actuator commands). Further, due to increased connectivity to the internet (both for access and control), several attack vectors have emerged that an adversary can utilize to exploit, disrupt, or damage the CPS. These *multi-domain interactions*, coupled with concerns of *security and resilience* make the task of analyzing the CPSs significantly challenging.

Owing to these complex interactions among system components and many possible combinations and manifestations of attack vectors, analysis of CPSs is usually performed using integrated simulations of different CPS components. However, this requires supporting time-dependent data exchange and time-synchronization between simulators. We rely on the IEEE High-Level Architecture (HLA) standard [2] for coordinated distributed simulations. In addition, due to performance reasons or unavailability of high-fidelity simulation models, many attacks and phenomena are not supported in simulations, thus requiring use of real, physical hardware for their emulation.

The tight coupling between physical and cyber aspects of the systems requires joint evaluation of CPS security and resilience. As shown in Figure 1, a testbed must enable the user to not only model the integrated system-of-systems, but also to systematically deploy various attacks on the integrated simulations. Our testbed, called the SecUre and Resilient Cyber-Physical Systems (SURE), uses a model-based integration approach, where models are used not only for the system modeling, but also for their configuration, parameterization, integration, and execution. Further, we created a library of attacks in cyber, physical, and hardware domains, from which the experimenter can pick any number of them and deploy systematically in the integrated simulations. We enable attacker-defender games to analyze the system's performance with different attacks as well as defense (and mitigation) strategies. Multi-stage games allow modeling of counter attacks and counter-counter attacks. Furthermore, Courses-of-Action (COA) enables experimenters to design workflow-like scenario models to perform a variety of *what-if* analyses in a systematic manner.

The complex nature of the real-world CPSs requires their analysis with many different configurations, parameters, and workflows, which can require significant computational infrastructure. A cloud-based experimentation backend can be used for testbed scalability. In SURE, a web-browser is used not only for metamodeling, system modeling, and COA and experiment designs, but also for the cloud experiment integration and display of live experiment results. Taken together, these capabilities provide a robust platform for analyzing the security and resilience of complex CPSs.

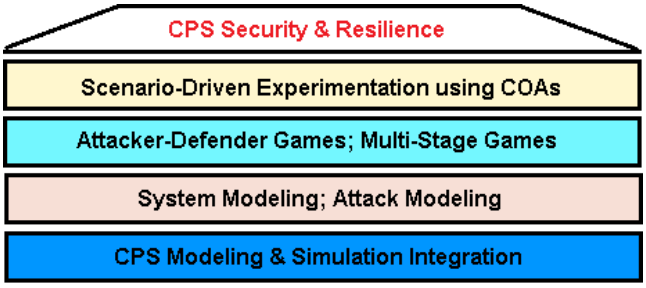

Figure 1: Capability Layers of CPS Evaluation Testbed

Security and resilience of CPS has attracted significant attention in many areas such as medical devices [16], automobiles [17], and transportation systems [18]. Owing to the heterogeneity and complexity of CPSs, their security methods are also highly diverse [19]. Several testbeds have been proposed for security assessment of CPS applications such as power and SCADA systems [20] [21] [22]. In contrast, our testbed provides unique capabilities such as web-based collaborative metamodeling, modeling of CPS, cloud computing platform for experimentation, attacker-defender and multi-stage games, scenario-driven experimentation using courses-of-action, and multiple layers of abstraction for modeling CPSs.

The rest of the paper is organized in the following manner. Section 2 paints the overall testbed architecture. Section 3 describes the hardware-in-the-loop simulation. Section 4 details the attack libraries in the testbed and how attacks can be deployed to carry out attacker-defender games. Section 5 presents the Courses-of-Action (COA) modeling. Experiment results are provided in Section 6 and we conclude the paper in Section 7.

2 TESTBED ARCHITECTURE

Our testbed focusses on providing modeling and experimentation tools to enable system designers and security researchers to be able to analyze security and resilience of CPSs. We use a web-based metamodeling and modeling tool, WebGME [4] for designing systems and scenarios, and configuring and executing experiments. WebGME uses a central server that can be accessed by multiple clients to enable collaborative modeling via the web. Additionally, the tool provides detailed tracing of modifications to help modelers with design evolutions by tracking change history and permitting branching and rollbacks like in a distributed source-code repository. We use road transportation networks as the application domain for our testbed, but all our tools are generic and have been used in many other domains such as power distribution and command-and-control simulations.

The testbed architecture is shown in Figure 2. WebGME is used to model the system and configure the simulation tools used for different aspects of the modeled CPS. The different simulation tools are integrated for timed data exchange and time-synchronized execution using the framework Command and Control Wind Tunnel (C2WT) [5] [6]. C2WT supports integration of a variety of simulation tools such as Matlab/Simulink [7], OMNeT++ [8], CPNTools [9], SUMO [10], TrainDirector [11], and Gridlab-D [12].

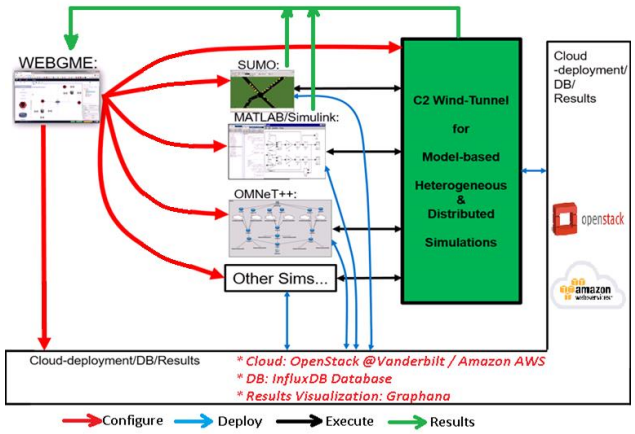

Figure 2: Testbed Architecture

As shown in Figure 2, we hosted the C2WT integration and execution platform in a cloud environment using OpenStack [23].

Additionally, when integrated simulations are executed in the C2WT platform the live experiment results are populated in a streaming InfluxDB database [24]. These results are then queried by the WebGME tool to display live charts of the experiment results as the simulations are executed. One key feature of our testbed is that it can execute many variations of the simulation experiments in parallel, limited only by the available cloud resources. This is crucial for analyzing CPS, which requires analyzing many configurations, parameters, and workflows.

Modeling of the CPS, both in WebGME and in C2WT, is based on Model-Integrated Computing (MIC) [13] that focuses on formally representing the system components, their interactions, and rules for composition and configuration. A metamodel is a Domain-Specific Modeling Language (DSML) designed for a particular application domain. For example, the metamodel in SURE captures the modeling of CPS systems and scenarios with a focus on security and resilience analysis, whereas in C2WT it focusses on distributed simulation integration.

One of the fundamental aspects of the testbed is the capability to analyze the CPS under a variety of attacks. The attacks currently supported and how they are used is described in Section 4. Further, the testbed allows detailed scenario modeling for performing what-if analyses using Courses-of-Actions (COAs) – described in detail in Section 5. In addition, the attack library and COAs use an integrated hardware-in-the-loop testbed to run hardware elements and deploy attacks in the hardware.

3 HARDWARE-IN-THE-LOOP SIMULATION

Many attacks and phenomena are not analyzable in simulations due to performance reasons or unavailability of high-fidelity simulation models, thus requiring use of real, physical hardware for performing Hardware-in-the-loop (HIL) simulation. However, the HIL platform must be connected with a distributed simulation platform, which provides scalability and time-synchronization needed for complex distributed simulations. Our HIL platform is comprised of two parts: a hardware-in-the-loop testbed and the C2WT distributed simulation environment. This allows for taking advantage of both the scalability of the C2WT with the fine-tuned ability to analyze CPS controller behavior on real emulated hardware consistent with the platforms deployed in the field.

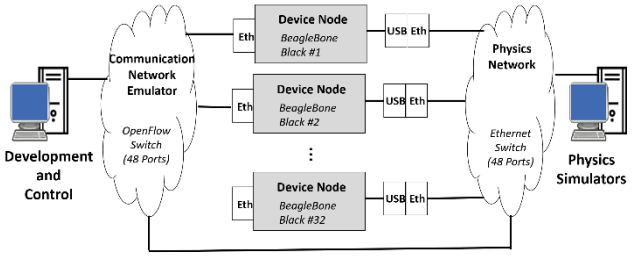

Figure 3: Hardware-In-The-Loop Platform

As illustrated in Figure 3, the HIL testbed is comprised of 5 different components. These include the development system where CPS control software is developed, the CPS nodes which consist of embedded computing boards consistent with operating platforms in the field, a software defined networking interface that enables controlling various communication parameters and protocols through the network, a Physics simulator serving as the physical plant, and a physical network connecting embedded computing nodes with the simulator interface.

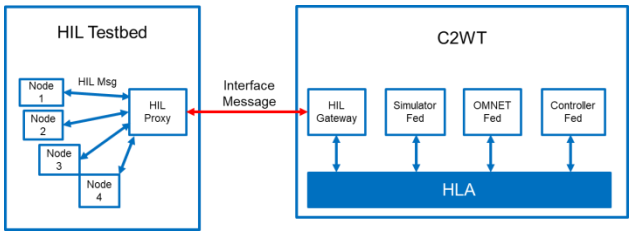

Figure 4: HIL Integration with C2WT

The HIL testbed also has an interface (see Figure 4) to connect the emulated software in the hardware with the simulated software in C2WT. The interface protocol communication utilizes Google Protocol Buffers [14] – a language and platform neutral extensible mechanism for serializing data – for formatting custom messages, and the ZeroMQ API [15] for transmitting and receiving messages throughout the network.

The integration interface has two components, viz. HIL Proxy and HIL gateway. The role of HIL Proxy is to serve as the interface between the embedded computing nodes on the HIL testbed and the simulation environment. As such, this proxy mechanism receives sensor information from each HIL node as well as sends custom commands to each respective node to adjust behavior. The role of the HIL gateway is to serve as an interface between C2WT simulators and the controller code in the HIL testbed. This can include communication between controllers and sensors defined in C2WT with controllers in the HIL testbed, as well as receiving controller commands from respective HIL nodes.

As the simulator interface is defined in C2WT as a simulation, the gateway is additionally responsible for serving as an interface for HIL node controllers to interact with the physical plant simulator. In addition, two message types are used, viz. HIL messages and interface messages. HIL messages correspond to internal messages in the HIL testbed such as internal controller communications or commands. However, when communication needs to be established with the C2WT environment such as obtaining sensor values or sending actuation commands to the simulator, interface messages are utilized for transmission.

4 ATTACK MODELING & CYBER GAMING

As a part of our testbed, a cyber-attack library exists for developing modular and reusable attack sequences. These attacks represent atomic actions that can be chained together with associated timing parameters to develop complex sequences of attacks on CPS. The attack library consists of three groups of attacks: cyber-attacks, physical attacks, and hardware attacks. Cyber-attacks compromise cyber components (e.g., network infrastructure) and message communications. Physical attacks compromise the physical road infrastructure such as vehicle crashes, lane closures, or traffic light failures. Hardware attacks are implemented in the real hardware.

Table 1: Testbed Attack Libraries

Attack type	Description
CYBER ATTACKS	
DOS	Disable a controller, or network component
Disable Network	Disable communication within the network
Delay	Delay packets when they are routed between components
Integrity	Change packet values before they reach destination
Data Corruption	Make packets unreadable
Replay	Retransmit packets
Out of Order	Sent transmitted packets in the wrong order
Network Filter	Filter out traffic between given source & destination subnets
Sniffer	Listen to communicated traffic
Routing Table Manipulation	Redefine the network routes
PHYSICAL ATTACKS	
Lane Closure	Close a road or lane of a road
Vehicle Failure	Cause a vehicle to stall
Traffic Light Failure	Cause a traffic light to stop operating
Vehicle Crash	Cause vehicles to crash within the simulation
HIL ATTACKS	
DOS/DDOS	Transmit bulk traffic from multiple nodes
Side Channel	Reverse engineer components based on behavior
Spoofing	Transmit fake messages

As shown in Table 1, the current cyber-attack library includes several key cyber-attacks such as denial of service, delay, corruption, and integrity attacks. Additionally, each of these contains parameters to customize the attack. For example, for an integrity attack – that manipulates message level network packets – the parameters are various fields of the message. Integrity attack updates messages in the simulation by modifying the message's fields. The key attacks in the physical attack library include lane closures, vehicle failures, traffic light failures, and vehicle crashes. These attacks interact directly with the behavior of the physical plant simulator (SUMO). Consistent with the cyber-attack instances, each of these attacks has editable parameters that allow each generic model to become a unique implementation instance (e.g., editable lane field in a lane closure attack is used update SUMO configuration to ensure the lane as untraveled). Finally, the key hardware attacks include side channel, [distributed] denial of service (DDoS), and spoofing attacks. These attacks allow observing the effect of multiple collaborating attack nodes on the outcome of an attack. This is vital in DDoS attacks where the magnitude of system degradation is directly proportional to the amount of attacking nodes.

Using the attack libraries we created several cyber-gaming test scenarios for CPS security and resilience. Below are a few such scenarios using a road transportation application domain (with traffic sensors, traffic lights and controllers, etc.):

1. ***Observation selection***: A set of traffic sensor locations are to be selected from many possible locations such that the chosen set is resilient to DoS attacks on a subset of them. It uses regression analysis to learn the covariance matrices model among the sensors and uses the learned model to predict traffic parameters. The integrated simulation is run on C2WT platform and the actual traffic flows are compared against the predicted values. The scenario allows attacker-defender gaming by allowing defenders to set locations of sensors and the attacker to attack a subset of those locations. The scenario can be used by multiple players at the same time via the WebGME modeling tool.
2. ***Resilience monitoring and control***: This scenario enables analyzing resilient control algorithms. After the defender has designed the initial traffic light control parameters, the attacker uses integrity cyber-attack to alter the traffic light phases and/or phase durations so as to maximize the average travel time of vehicles. Different control algorithms can be compared against different attack schemes.
3. ***Resilient architectures***: This scenario was designed to incorporate attack parameter values generated from real hardware based attack deployments. Using these realistic attack parameters in the integrated simulation environment provides more accurate analysis of CPS architectures.
4. ***Hierarchical controller***: It enables system-level resilience studies by incorporating system-level control algorithms to monitor and reconfigure low-level control algorithms. For example, this scenario could be used to study resilient algorithms that evolve in depth and complexity depending on how threats emerge and evolve.
5. ***DoS with Hardware-in-the-loop***: It allows HIL simulations by integrating the HIL testbed with C2WT.

5 COURSES-OF-ACTION ANALYSIS

Courses-of-Actions (COAs) allow for modeling scenarios that describe detailed what-if analysis or multi-stage attacker-defender gaming. As shown in Figure 5, the basic idea of COAs is to enable analysis of integrated simulations along with *dynamic behavior* to exercise the same system models with many different behavioral or scenario workflows.

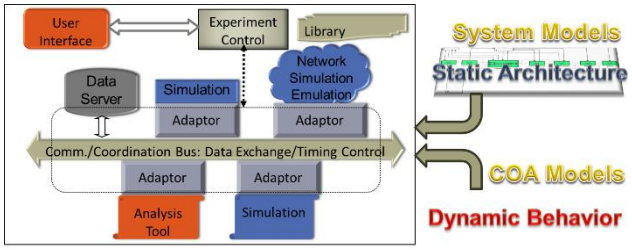

Figure 5: Courses-of-Action (COA) for Dynamic Behavior

COAs are workflow-like scenario models that are created using several atomic *actions* and atomic *outcomes* (i.e. triggers) such as time, an event, or system outputs. Additionally, the COAs contain a variety of COA elements (see Table 2), such as forks and random durations, which can be used for elaborate strategy or game planning. A COA model is a Directed Acyclic Graph (DAG) that is created by connecting atomic nodes with directed edges. A generic COA *Orchestration Engine* is used to execute one or more COAs in parallel as individual atomic elements of COAs become enabled (after preceding node's execution).

Table 2: COA Atomic Elements

COA Element	Description
Synchronization Point	Represents absolute time-point from beginning of simulation. All incoming branches must wait until this time-point has been reached.
Action	An interaction that must be sent out by the COA Orchestration Engine as soon as the action point is reached. The parameters of the interaction can be specified.
Outcome	Represents the type of interaction that the COA Orchestration Engine must wait to arrive before it can proceed.
Fork	A branching element with the following sementics: All branches following this element are executed in parallel.
Probabilistic Choice	Chooses only one succeeding branch based specified probabilities of outgoing branches.
AwaitN	Waits on a given number of incoming branches to finish before letting the COA Orchestration engine proceed.
Duration	Represents the time the COA Orchestration Engine delays the execution once the duration element is reached.
Random Duration	A duration that is randomly distributed using a uniform distribution.
Outcome Filter	Filter based on the values of the parameters of the received interaction. Different outgoing branches can be executed based on different values of parameters.
Terminate COA	When reached, the COA execution is terminated.
SimEnd	When reached, the entire simulation of the federation is terminated.

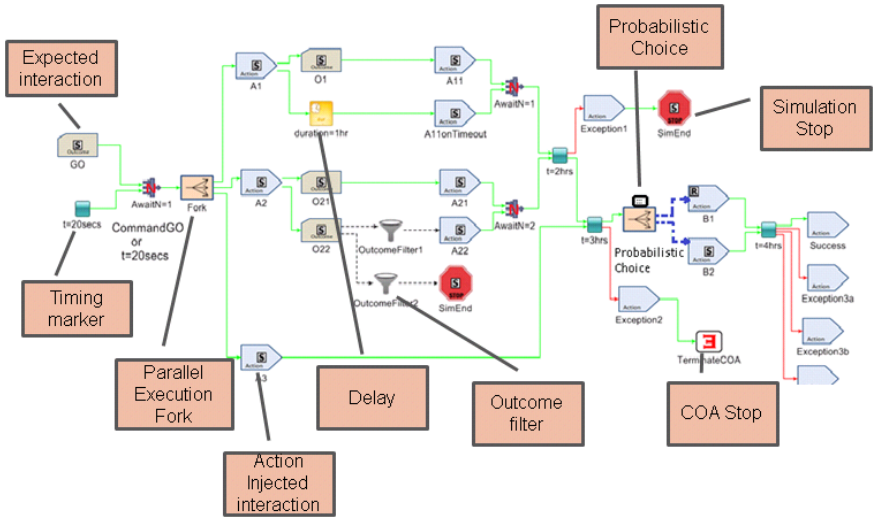

Figure 6: Illustrative Example of a COA

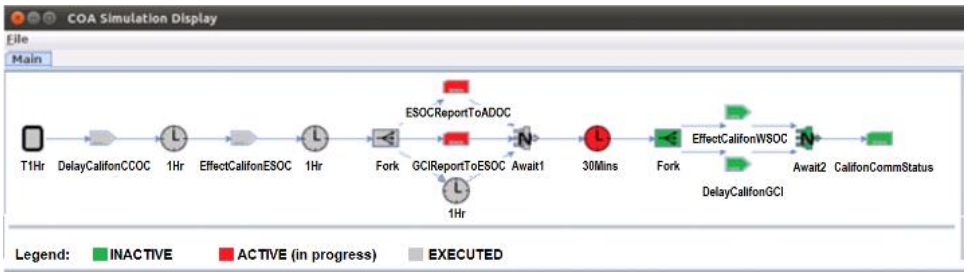

Figure 7: COA Status Display

Figure 6 illustrates an example COA model. Note that the *action* element injects new information into the running simulation. Whereas, the *outcomes* are the observation patterns that must match in order for the COA execution to proceed further along the branch. Execution status of COAs is also made available by the testbed. As shown in Figure 7, the status display

shows different color for different statuses of nodes, viz. red for currently executing nodes, gray for nodes that have finished execution, and green for inactive nodes.

In addition, the testbed allows for the grouping of COA models and to automatically select combinations of COAs from different COA groups. This is referred to as *design of experiments* in the testbed. As an example, if the defender created 6 different defense strategies as COA models in a defender COA group and the attacker created 5 different attack strategies as COA models in the attacker COA group, then choosing the two COA groups for experimentation will automatically execute 5 x 6 = 30 different combination of scenarios. Such a capability is useful for cyber gaming as well as multi-stage attacker-defender games for the security and resilience analysis of CPS's.

6 EXPERIMENTAL RESULTS

Case-study 1: Observation selection scenario: Figure 8 shows a road transportation network in the background overlaid with a cyber communication network topology. The traffic sensors are the circles marked with the letter 'S'. The red colored sensors are those under DoS attack, while blue colored sensors are operational. A set of sensors need to selected such that even when a subset of those are under cyber-attacks, the estimation of traffic flow at the magenta colored location will be close to acceptable compared to the real traffic flow in the simulation.

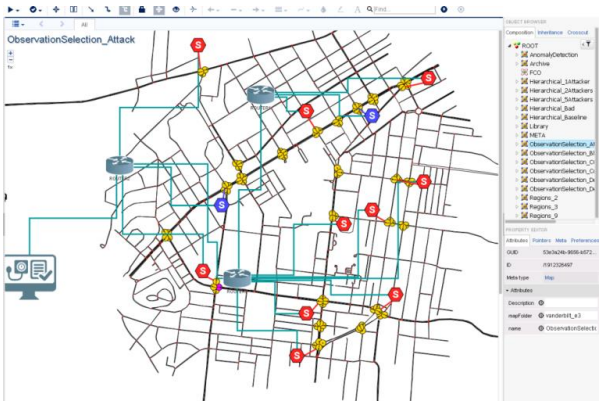

Figure 8: Observation Selection Experiment Scenario

We simulated the scenarios where none of the sensors were attacked and the one shown in Figure 8 with several sensors under DoS attack. The results from these two variations are recorded by C2WT in an InfluxDB database, which were pulled live by WebGME and shown to the user (see Figure 9). Also, when the experiments are completed, the Root Mean Square Error of the measured traffic densities is compared against the actual flows for both cases, viz. when no sensors were attacked, and when some sensors were attacked. In our experiments, we found that when several sensors were attacked, the RMSE increased from 10.546 to 20.0068. Our testbed also computes the graphs of predicted traffic densities vs observed (in simulation) traffic densities.

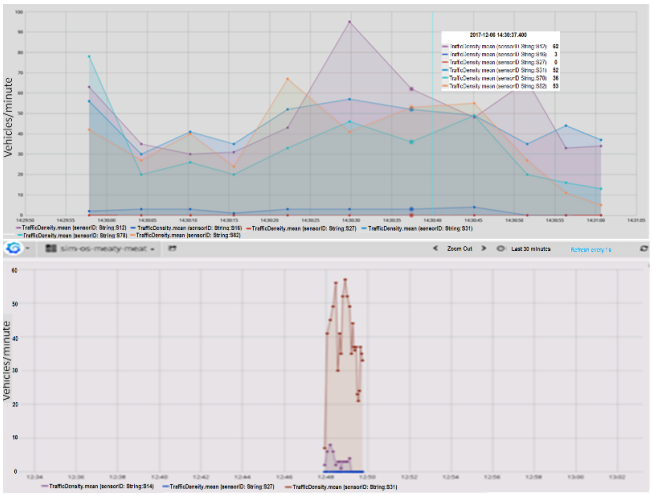

Figure 9: Live Experiment Results in Web-Browser

Case-study 2: Hardware-in-the-loop scenario: A traffic control study (see Figure 10) is used with 9 traffic light controllers within a university campus routing vehicles efficiently throughout the area. Additionally, there is a level 1 trauma center near the center of campus making it crucial for ambulances and other emergency vehicles to travel through this area with as little traffic as possible. The intersection traffic light controllers are split into two areas: intersections simulated in C2WT and controllers emulated in the HIL testbed.

Each intersection in C2WT includes induction loop sensors that detect traffic density at each direction. This information is fed to the traffic light controller that then executes a queue based controller algorithm to optimize traffic flow through the intersection. However, the induction loop sensors communicate wirelessly to the controller making them vulnerable to man in the middle attacks that spoof or modify routed packets before they reach the controller.

Each intersection emulated in the HIL testbed is controlled through a fixed time schedule, preventing the possibility of communication based integrity attacks, but also optimizing traffic less efficiently than the queue based traffic light controllers. The fixed time controllers are however prone to DoS attacks based on power disruptions or physical attacks. The attack surface is defined as follows: an attacker can perform an integrity attack at an intersection in C2WT area and edit induction loop sensor packets to show fewer cars at the intersection. Additionally, the attacker can perform a DoS attack on an intersection within the HIL area, to disrupt the path of critical routes.

The attacker tries to maximize the average vehicle trip duration in the road network, as well as minimize the average speed of traffic through the area. The results of the experiment are shown in Figure 11. During the baseline scenario where no attacks are observed, the average trip duration is approximately 284 seconds with an average trip speed of 24.45 miles per hour. However, when the attacker performed integrity and denial of service attacks on intersections, traffic flows were affected significantly throughout the area. In this case, the average vehicle trip duration increases to 512 seconds, almost double the original

trip duration value. Furthermore, the average speed decreases to approximately 21.22 miles per hour, a 13% reduction compared to the baseline scenario.

Figure 10: HIL Integration Scenario

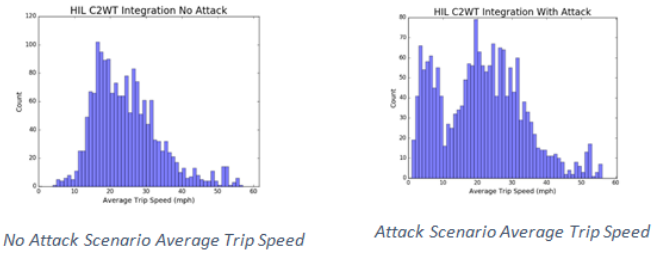

Figure 11: Average Speed Comparison

7 CONCLUSION

Cyber-physical systems are becoming wide-spread in various parts of the economy including the critical infrastructure. Secure and resilient CPSs are vital to avoid disruptions and damages. Evaluating CPS security and resilience is of extreme importance, yet is a highly challenging task. Our testbed facilitates this evaluation using model-based simulation integration, web-based collaborative system and experiment modeling, and cloud experiment execution. The testbed provides modular and extensible attack libraries for the physical, cyber, and hardware attacks. Also, the testbed provides modeling and experimentation with Courses-of-Action (COAs) that enable analysis with different what-if and gaming scenarios. The COAs provide an intuitive means to enable system analysts to perform many scenario-driven experiments over the same integrated simulation. These sophisticated modeling and experimentation capabilities make our testbed a robust, powerful, and scalable platform for analyzing the security and resilience of CPSs.

ACKNOWLEDGMENTS

This work was supported in part by Air Force Research Laboratory under Award FA 8750-14-2-0180, National Science Foundation under Grant CNS-1238959, and National Institute of Standards and Technology under Award 70NANB17H266. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] R. Baheti and H. Gill, "Cyber-physical systems," The impact of control technology, vol. 12, pp. 161–166, 2011.
[2] "IEEE Std 15162010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules", pp. 1–38, 2010.
[3] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a science of cyber–physical system integration," Proceedings of the IEEE, vol. 100, no. 1, pp. 29–44, 2012.
[4] M. Maroti, R. Kereskenyi, T. Kecskes, P. Volgyesi, and A. Ledeczi, "Online collaborative environment for designing complex computational systems," in The International Conference on Computational Science (ICCS 2014). Elsevier Procedia, 06/2014 2014.
[5] Hemingway, G., H. Neema, H. Nine, J. Sztipanovits, and G. Karsai, "Rapid Synthesis of High-Level Architecture-Based Heterogeneous Simulation: A Model-Based Integration Approach", SIMULATION, vol. March 17, 2011 0037549711401950, no. March 17, 2011, Online, Simulation: Transactions of the Society for Modeling and Simulation International, pp. 16, 03/2011.
[6] Neema, H., H. Nine, G. Hemingway, J. Sztipanovits, and G. Karsai, "Rapid Synthesis of Multi-Model Simulations for Computational Experiments in C2", Armed Forces Communications and Electronics Association - George Mason University Symposium, issue Critical Issue in C4I, Lansdowne, Virginia, 05/2009.
[7] "Matlab/simulink." [Online]. Available: https://www.mathworks.com/products/simulink.html
[8] A. Varga, "The OMNeT++ discrete event simulation system. in: Proceedings of the european simulation multiconference (ESM'2001)," Prague, Czech Republic, 2001.
[9] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri nets and CPNtools for modelling and validation of concurrent systems," International Journal on Software Tools for Technology Transfer, vol. 9, no. 3-4, pp. 213–254, 2007.
[10] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO – Simulation of Urban MObility: An overview," in Proceedings of the 3rd International Conference on Advances in System Simulation (SIMUL), 2011, pp. 63–68.
[11] "Traindirector." [Online]. Available: http://www.backerstreet.com/traindir/
[12] D. P. Chassin, J. C. Fuller, and N. Djilali, "Gridlab-d: An agent-based simulation framework for smart grids," Journal of Applied Mathematics, vol. 2014, 2014.
[13] Sztipanovits J., Karsai G.: Model-Integrated Computing. In: IEEE Computer 30, 1997, pp. 110-112.
[14] Varda, Kenton. "Protocol buffers: Google's data interchange format." Google Open Source Blog, Available at least as early as Jul 72 (2008).
[15] Hintjens, Pieter. ZeroMQ: messaging for many applications. " O'Reilly Media, Inc.", 2013.
[16] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel, "Security and privacy for implantable medical devices," IEEE Pervasive Computing, vol. 7, no. 1, 2008.
[17] J.-P. Hubaux, S. Capkun, and J. Luo, "The security and privacy of smart vehicles," IEEE Security & Privacy, vol. 2, no. 3, pp. 49–55, 2004.
[18] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Enhancing security and privacy in traffic-monitoring systems," IEEE Pervasive Computing, vol. 5, no. 4, pp. 38–46, 2006.
[19] J. Giraldo, E. Sarkar, A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," IEEE Design & Test, 2017.
[20] C. Davis, J. Tate, H. Okhravi, C. Grier, T. Overbye, and D. Nicol, "SCADA cyber security testbed development," in 38th North American Power Symposium (NAPS 2006), 2006, pp. 483–488.
[21] T. Kropp, "System threats and vulnerabilities [power system protection]," IEEE Power and Energy Magazine, vol. 4, no. 2, pp. 46–50, 2006.
[22] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri, "A testbed for analyzing security of SCADA control systems (TASSCS)," in 2011 IEEE PES Innovative Smart Grid Technologies (ISGT),, 2011, pp. 1–7.
[23] "Openstack cloud." [Online]. Available: www.openstack.org
[24] "InfluxDB database." [Online]. Available: www.influxdata.com.