A comprehensive approach, and a case study, for conducting attack detection experiments in Cyber–Physical Systems

Sabaliauskaite, G., Ng, G. S., Ruths, J. & Mathur, A.

Original citation & hyperlink:
Sabaliauskaite, G, Ng, GS, Ruths, J & Mathur, A 2017, 'A comprehensive approach, and a case study, for conducting attack detection experiments in Cyber–Physical Systems', Robotics and Autonomous Systems, vol. 98, pp. 174-191.
https://dx.doi.org/10.1016/j.robot.2017.09.018

DOI 10.1016/j.robot.2017.09.018
ISSN 0921-8890

Publisher: Elsevier

A Comprehensive Approach, and a Case Study, for Conducting Attack Detection Experiments in Cyber Physical Systems

Giedre Sabaliauskaite, Geok See Ng, Justin Ruths, and Aditya Mathur

Abstract

Several methods have been proposed by researchers to detect cyber attacks in Cyber-Physical Systems (CPSs). This paper proposes a comprehensive approach for conducting experiments to assess the effectiveness of such methods in the context of a robot (Amigobot) that includes both cyber and physical components. The proposed approach includes a method for performing vulnerability analysis, several methods for attack detection, and guidelines for conducting experimental studies in the context of cyber security. The method for vulnerability analysis makes use of the Failure-Attack-CounTermeasure (FACT) graph. The experimental study to evaluate methods for attack detection comprises of three experiments. These methods have been implemented and evaluated, within and across all three experiments, with respect to their effectiveness, detection speed, and durability for injection, scaling, and stealthy attacks. The proposed guidelines define key phases and artifacts for conducting such experiments and are an adaptation of those used in Software Engineering.

Index Terms

cyber-attacks, cyber-physical systems, robots, security, CUSUM method, intelligent checker, FACT graph.

I. Introduction

Cyber-Physical Systems (CPSs), which integrate both cyber and physical worlds, are continuously expanding their application areas to various robots, smart devices, transportation, and critical infrastructures among others. Consequently, the amount of CPS research is increasing,

Giedre Sabaliauskaite, Geok See Ng, and Aditya Mathur are with the Singapore University of Technology and Design, 8 Somapah Road, Singapore 487372. Justin Ruths is with University of Texas at Dallas, Richardson, TX, 75080-3021, USA

in particular in the area of cyber-security, as CPS are vulnerable to cyber-attacks, which might have devastating consequences.

Numerous experimental studies into the security of CPSs have been reported in the literature, such as e.g. [1], [2], [3], [4]. However, all these studies are described in different ways, since, to the best of our knowledge, the are no guidelines or unified procedure for reporting on cyber-security experiments. This makes it difficult to compare experimental results and to replicate the studies.

The goals of this paper are twofold: firstly, to propose and approach for conducting CPS cyber-security empirical studies; secondly, to validate the proposed approach by applying it to an experimental study.

The proposed approach includes the method for CPS vulnerability analysis, several attack detection methods, and the guidelines for conducting CPS security experiments. A graph, named the Failure-Attack-CounTermeasure (FACT) graph [5], was used for CPS vulnerability analysis and attack identification. Furthermore, seven anomaly detection methods have been proposed, implemented, and experimentally evaluated. These methods could be used for detecting various attacks in different types of CPSs. Finally, the guidelines for CPS security experimentation have been defined based on the guidelines for case study research in software engineering, proposed by Runeson et al. [6].

The experimental study comprised of three experiments, which investigated the vulnerabilities of Cyber Physical Systems (CPS) to cyber-attacks, and the performance of anomaly detection methods in detecting the attacks on sensor data. Specifically, the CPS used in this study was a commercially available robot known as the Amigobot [7].

Denial of Service (DoS) and false data injection attacks are two the most common attack types on CPS [8]. DoS attacks prevent new data from reaching either controllers or sensors [8], while false data injection attacks send corrupted data to sensors or actuators [9], [10]. False data injection attacks are more subtle and difficult to detect as compared to DoS attacks [8], [10].

This study focuses on false data injection attacks that result in deliberate sensor data corruption. A survey of data corruption in critical infrastructure CPS [9] describes various anomaly detection approaches for detection of corrupted data. Statistical and behavioral approaches can be used for this purpose. Statistical approaches detect attacks by identifying irregularities in the data, while behavioral approaches are aimed at detecting anomalous behavior of the system [9]. Although one could test anomaly detection algorithms in simulation environments [10], the need for doing

so in the context of real-systems is considered important and has been in several experiments using a variety of testbeds [1], [2], [3].

This paper proposes and implements statistical approaches. There are at least three types of statistical approaches: rule-based, estimation-based, and learning-based [9]. Rule-based approaches are the simplest form of anomaly detection based on acceptable lower and upper limits of data. Any value outside this range is an anomaly [9]. Estimation-based detection approaches use estimated data to detect anomalies, while learning-based approaches utilize data mining, clustering and classification algorithms [9].

CUSUM method [11] is an estimation-based approach, used in this study. It detects changes in mean values of the process. Sequentially accumulated data values that are higher than the mean value are observed under normal CPS operation. The CUSUM value is compared to a threshold, and the anomaly is detected if CUSUM exceeds the threshold [11]. The CUSUM method has been previously implemented in several studies: detection of network attacks, such as wormholes, jamming, etc., in wireless sensor networks [12]; detection of D0S attacks in networks [13]; false data detection in a simulated chemical plant [10] and a water treatment testbed [14].

Intelligent Checker (IC) based method is one of the rule-based approaches, implemented in this study [15]. ICs are smart sensors primarily used to detect attacks on CPS and alert the operators [15]. ICs monitor the status of the controlled physical process and raise an alarm when the process measurements violate predefined constraints. In this study, the measurements of ICs were used to detect sensor attack based on the average difference between ICs' and sensors' measurement.

The remaining rule-based attack detection approaches, implemented in this study, used average robot's distance data in their attack detection algorithms. A study of using rule-based and other statistical approaches for intrusion detection in SWaT testbed [3] has been reported in [4].

Three types of cyber-attacks on sensor measurements were implemented: false data injection, scaling, and stealthy attacks. In the injection attacks, sensor measurements were biased through addition, while in scaling attacks the bias was through multiplication. Stealthy attacks are attacks designed by attackers to avoid detection by utilizing knowledge of not only the system's model but also the parameters of detection method, such as detection threshold. Three subtypes of stealthy attacks - surge, bias, and geometric [10] - have been implemented to avoid detection by the CUSUM method. Stealthy attack detection has been previously performed in a simulated chemical plant [10] and a water treatment testbed (SWaT) [3].

The first experiment of this study, referred to as EXP1, was designed to test Amigobot as a possible testbed for CPS security and to investigate the performance of several methods in detecting attacks on Amigobot's sensors. Simulation and three actual Amigobots were used for this experiment. Injection, as well as scaling attacks, have been implemented. Furthermore, in addition to pre-programmed attacks, actual, man-in-the-middle attacks have been executed.

The goal of the second experiment, referred to as EXP2, was to further compare the performance of several attack detection methods in detecting and responding to injection attacks.

Finally, the third experiment, referred to as EXP3, was focused on stealthy attacks: investigation of possible damage of stealthy attacks on Amigobot, and the performance of seven methods in detecting stealthy attacks in simulation and an actual robot.

The remainder of the paper is organized as follows. Section II includes the preliminaries. State of the art is described in Section III. Experimental setup of experiments EXP1-EXP3 is explained in Section IV, while experiment execution and results are described in Section V. The discussion is presented in Section VI. Finally, Section VII concludes the paper.

II. Preliminaries

A. System description

The Amigobot robot, hereafter referred to as the robot, is used in this study. It is a general purpose differential-drive standard robot platform. The following capabilities of the robot are used in the project and are available in many other forms of mobile robots: ultrasonic distance sensors (sonars) for obstacle sensing; communication with user software via Wibox wireless transmitter [16]; user software communicates with the robot controller via SDK to request robot motion and receive updates on robot position and status; SDK toolkit running in Windows with programming libraries which can be used by Python; and ROS (an open-source robot operating system and is available for download at ROS.org website) interface. These capabilities provide solutions that are suitable to the basic needs of our research.

The sonar firing rate of the robot is set to 0.05 seconds. In user software, keeping track of robot update timing is critical, especially for real-time application. Hence the robots update loop is fixed at a time step of 0.1 seconds. These settings work hand in hand. The setting of the time step of 0.1 seconds is to make sure sonar reading period is enough for each sonar update. The next lower setting of time step may overlap with sonar update period of 0.05 seconds and may cause missing reading issue.

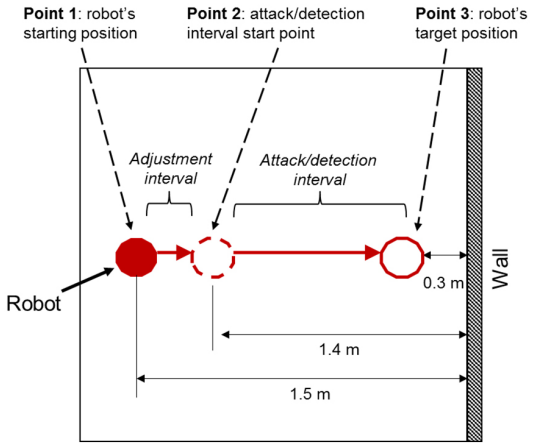

Fig. 1. Controlled system.

The controller in the robot used measurements of one frontal sensor (i.e. sonar) to determine its current position. Initially, the robot was placed at a distance of 1.5 meters from the wall, as shown in Figure 1. It then moved straight ahead in the direction of the wall at a variable speed proportional to the distance from the wall, i.e., the smaller the distance the lower the speed, until it reached the desired distance of 0.3 meters (Figure 1). After the robot reached its destination, it maintained the desired distance until the end of the run.

The velocity of the robot v_k (measured in m/s) at time k is computed as

$$v_k = -0.25(0.3 - \tilde{y}_k), \tag{1}$$

where 0.3 is the target distance (0.3m from the wall) and the constant of proportionality was determined empirically to be -0.25 based on the robot's performance and the desired duration of the experiment.

The Amigobot has limitations regarding measurements at short distances. Through extensive testing of the robot, it was discovered that it may become unstable when it reaches a distance smaller than 0.2m from the wall. Furthermore, if the distance from the wall is less than 0.15m, the robot might move forward by itself and consequently crash into the wall even without any component failures or cyber-attacks.

To avoid damage to the robot in the catastrophic cases where it crashed into the wall, we used a lightweight foam wall instead of an actual wall, which could be easily moved by the robot without causing any damage to the robot.

Furthermore, the robot has a warm-up period, and its speed and direction varies across runs; in some runs, the robot starts moving more slowly compared to other runs; in other cases, it moves slightly backward. To avoid false attack detection during the warm-up period, an adjustment interval of 10 centimeters was introduced at the beginning of each run (see Figure 1). During this interval, no detection of attacks was performed to avoid false positives. Subsequent to this interval, robot behavior during different runs is nearly the same.

B. The CUSUM Method

Our study uses CUSUM approach in four attack detection methods: M3, M4, M6, and M7 (see Section III-B). CUSUM detects changes in mean values of the process, for identifying sensor attacks in the robot. Sequentially accumulated data values that are higher than the expected value are observed under normal CPS operation. The CUSUM value is compared to a threshold, and the anomaly is detected if CUSUM exceeds the threshold [11]. Expected sensor value can be computed either by using historical data (as in methods M3 and M4), or estimated based on system's physics model (as in methods M6 and M7) (see Section III-B). The data collected during experimental runs in a trusted environment is referred to here as *historical data*. The algorithms used for detection by M6 and M7 are non-parametric CUSUM algorithms, as they detect changes in mean values of the process without assuming any knowledge about the underlying data distribution [11].

The CUSUM approach relies on biasing the distribution of absolute errors between the expected output of a system, $\hat{y}_k$, and its actual output, $\tilde{y}_k$, to be less than zero on average. The biased absolute error at time step k is given by

$$z_k = |\hat{y}_k - \tilde{y}_k| - b, \tag{2}$$

where the bias parameter b is selected by computing the distribution of $|\hat{y}_k - \tilde{y}_k|$ over many samples and in a trusted environment so that the deviations are due to systematic errors in the model, and not due to an attack. The goal here is to ensure that the expected value (the mean) of the z_k distribution is less than 0, so the bias is selected slightly larger than the mean. The CUSUM statistic is the sum S_k of the z_k values, with a cutoff that keeps the statistic nonnegative, $S_k = (S_{k-1} + z_k)^+$, where $a^+ = a$, if $a \geq 0$, and zero otherwise. When the CUSUM statistic S_k surpasses a threshold τ, the operator of the CPS is notified of a potential attack. In an automatically controlled system, the operator might be the response mechanism

that must defend the CPS against the attack, possibly using a model of the system. The value of τ is set empirically to reduce the number of false positives.

C. The Intelligent Checker

ICs are the smart sensors, which are implemented in parallel to the CPS control loop [17]. The control loop can be described as a work-flow of four main steps: *monitoring* – physical processes and environment are monitored by the use of sensors; *networking* – allows the data generated by sensors to be sent to the controller; *computing* – the data collected through monitoring is analyzed and further actions are determined by the controller (e.g. Programmable Logic Controller (PLC)); *actuation* – the actions determined during the computing phase are executed by the use of actuators. These steps form a control loop between controller, actuators, controlled process, and sensors, as shown in Figure 2.

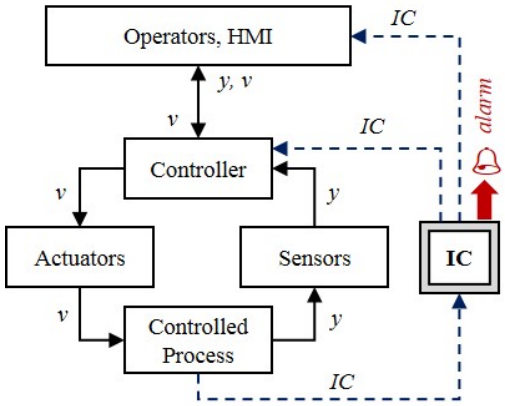

Fig. 2. Intelligent Checker, IC, and a control loop.

ICs are not part of any CPS control loop [17]. However, ICs may send data to one or more components of the control loop, e.g. controllers (as in Figure 2). ICs are aimed at monitoring critical parameters of the controlled process with a goal to identify cyber as well as physical attacks and failures in CPS. They may measure various parameters of the physical process, such as pressure, temperature, distance, etc.

IC consists of two elements: *IC sensor*, which measures selected parameters of the controlled process; *Decision logic*, which compares the measurements against predefined constraints. Decision logic element is connected to an alarm, which is activated if the constraints are violated

(see Figure 2). ICs have no incoming traffic from any portion of the communication network in the CPS, which prevents attackers from modifying IC sensor measurements. In the event where the communication channel between IC and other CPS components is compromised, an IC continues to be a useful entity as it generates audible or visible alarms when measurement constraints are violated.

In this study, the ICs are used by the detection method M5, where ICs measure robot's distance from the wall and to send these measurements to a controller. The controller compares IC's measurements to control loop's sensors' measurements in order to detect attacks. The robot has two front sensors (sonars), which measures the distance from the wall. One of the sensors (sonar 3) was used to control the robot – its measurements were used by a controller to compute control input; another sensor (sonar 2) was used as an IC.

III. State of the Art

A. Analysis of CPS Vulnerabilities to Cyber-Attacks

Nowadays, CPSs are vulnerable not only to accidental component failures or software errors (safety failures), but also to intentional (security) attacks, which could lead to fires, floods, chemical spills, potential crashes of vehicles, etc.

Furthermore, safety and security are interdependent. Very often they either complement or conflict each other [18], [19], [20]. There are at least four types of inter-dependencies [21]: 1) conditional dependencies: security is a condition for safety and vice versa; 2) reinforcement: safety and security countermeasures can strengthen each other; 3) antagonism: they can weaken each other; 4) independence: no interaction between safety and security. Thus, it is an important to analyze safety-security interdependencies when choosing security countermeasures to protect CPS from attacks.

What can we use to help us in identifying CPS vulnerabilities to cyber-attacks?

In [5], a Failure-Attack-CounTermeasure (FACT) graph has been proposed for CPS safety and security modeling. The FACT graph is based on safety and security standards ISA84 and ISA99 [22], [23]. It incorporates safety and security artefacts, and can be used for analyzing the vulnerabilities of the systems and selecting countermeasure set to provide a necessary level of protection.

We propose to use the FACT graph for analyzing Amigobot's vulnerabilities and identifying the interrelations between failures and attacks before the experiments, as shown in Figure 3.

After construction of the FACT graph, several important robot failure modes can be selected for implementation in experiments. During the experiments, the performance of security countermeasures is evaluated. After the experiments, the best performing security countermeasures could be added to the FACT graph to identify all attack and failures that are detected and/or mitigated using these countermeasures.

The FACT graph construction consists of the following four stages [5]: 1) fault trees construction to form a base of the FACT graph; 2) addition of safety countermeasures by attaching them to the related failures; 3) addition of attack trees to the graph; attacks, related to failures in a FACT graph, are attached to the corresponding safety failures; attack trees are incorporated into fault trees by the use of AND/OR gates, which indicate that a failure may be caused either by accidental failures, and/or by intentional attacks; 4) addition of security countermeasures to the FACT graph.

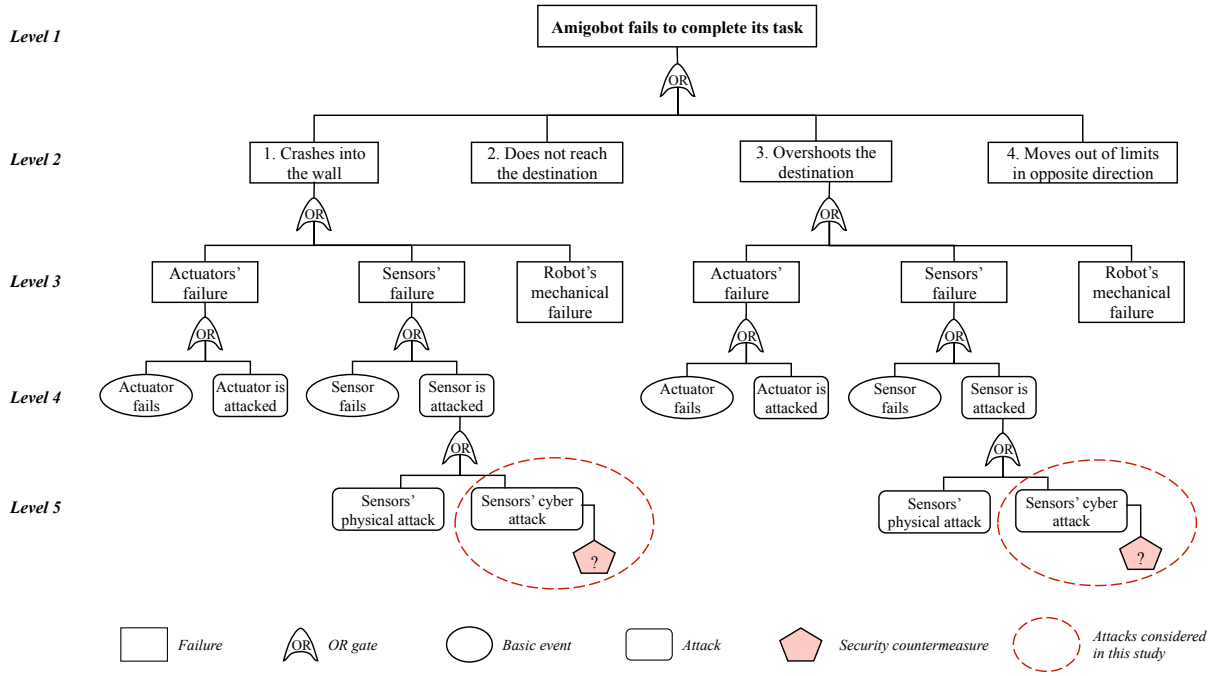

Fig. 3. FACT graph of the robot. Four high-level failures numbered 1 through 4 are considered.

We will explain the FACT graph's construction process using the Amigobot's example (see Figure 3). The construction of the fault trees starts with identification of a top-level undesired event. The top-level undesired event of Amigobot is failing to complete the task of reaching

the destination of 0.3m from the wall and maintaining this distance (see Figure 3 level 1). The following are the lower level undesired events that might lead to a top-level event (Figure 3 level 2):

1) Robot crashes into the wall;
2) Robot does not reach the destination;
3) Robot overshoots the destination (comes closer to the wall than expected);
4) Robot moves out of limits in opposite direction, and could possibly crash.

Four attackers' goals, G1-G4, can be identified based on the above failures: G1 – crash the robot into the wall; G2 – prevent the robot from reaching its destination; G3 – make the robot overshoot the destination; G4 – move the robot in the opposite direction with the intention to crash it into the opposite wall.

Attackers goals G1 and G3, which correspond to, respectively, failures 1 and 3 in Figure 3 (overshooting destination and/or crashing into the wall), were chosen for our study, as they were the most safety-critical. The FACT graph was then expanded to include lower level attacks and failures of the failures 1 and 3. As in Figure 3 level 3, failures 1 and 3 can happen due to failures of actuators, sensors, or the robot's mechanical system, or attacks on sensors and actuators. This study focuses on cyber-attacks on sensors, as shown in Figure 3 level 5, and evaluates the performance of several security countermeasures in detecting such attacks.

B. Methods for Detecting CPS Sensor Attacks

In total, seven anomaly detection methods, M1-M7, were developed and implemented in this study. These methods are not limited to Amigobot and can be used for sensor attack detection in various CPSs.

- M1, the *Average Distance Method*, compares sensor distance measurements $\tilde{y}_k$ at each time step k to the average distance from the wall at that time step, based on the historical data analysis (Eqn 3). For M1, τ is a threshold for sensor measurements at each time step, defined based on the standard deviation of historical data.

$$\tilde{y}_k = \begin{cases} \text{actual (true) distance if} \\ \qquad \bar{y}_k - \tau \leq \tilde{y}_k \leq \bar{y}_k + \tau \\ \text{attack (false) distance otherwise} \end{cases} \tag{3}$$

- M2, the *Average Distance Traveled Method*, compares the distance traveled between two measurement points $\tilde{u}_k$ (Eqn 4) to the average distance traveled $\hat{u}_k$ based on the historical data (Eqn 5). For M2, threshold τ is an average distance traveled between current and previous measurement point, and is defined based on historical data.

$$\tilde{u}_k = |\tilde{y}_{k-1} - \tilde{y}_k| \tag{4}$$

$$\tilde{u}_k = \begin{cases} \text{actual (true) distance traveled if} \\ \bar{u}_k - \tau \leq \tilde{u}_k \leq \bar{u}_k + \tau \\ \text{attack (false) distance otherwise} \end{cases} \tag{5}$$

- M3, the *Average Distance Change Method*, is a CUSUM method, which uses the average sensor value $\bar{y}_k$, computed using historical data (as in M1), as expected sensor value $\hat{y}_k$ in (Eqn 2).
- M4, the *Average Distance Traveled Change Method*, is a CUSUM method, which uses average distance traveled value $\bar{u}_k$, computed from historical data (as in M2) as expected distance traveled value. The biased absolute error at time step k is given by the following equation.

$$z_k = |\bar{u}_k - \tilde{u}_k| - b, \tag{6}$$

  where the bias parameter b is selected by computing the distribution of $|\bar{u}_k - \tilde{u}_k|$ over many samples.
- M5, the *IC-Based Method*, uses additional sensor's (IC's) measurements for anomaly detection. In this study, the second frontal sensor on the Amigobot was used as an IC. The IC measurements were used for sensor attack detection in the following way: at each time step k, a difference d_k between IC's measurements IC_k and sensor's measurements $\tilde{y}_k$ was computed:

$$d_k = |\tilde{y}_k - IC_k| \tag{7}$$

  If this difference $d(k)$ exceeded a predefined threshold, an attack was detected. Threshold was defined based on the average difference between sensor's and IC's measurements, which is calculated using historical data.
- The *Non-Parametric CUSUM Method*, M6, uses sensor measurements at previous measurement point along with the system's physics model to compute the expected sensor value.

A simple model was created to generate the expected (estimated) sensor values $\hat{y}_k$. In this case the estimation is based on the previous sensor measurement and the robot's velocity from (1),

$$\hat{y}_k = \tilde{y}_{k-1} - 0.1v_k = \tilde{y}_{k-1} - 0.025(0.3 - \tilde{y}_{k-1}), \tag{8}$$

where $\tilde{y}_{k-1}$ is the distance from the wall at the previous measurement point and the 0.1 value arises since sensor measurements and control actions occur every 0.1 seconds. In this method, the expected sensor value $\hat{y}_k$ depends not only on the robot's linear model, but also on the sensor measurements acquired in the previous step. Bias b was computed by biasing the z_k distribution to be less than zero, using (8) in (2).

- M7 is an *Alternative Non-Parametric CUSUM Method* developed for this study. As in Eqn 9, in M7 $\hat{y}_k$ depends on the previous estimate $\hat{y}_{k-1}$ rather than the previous sensor value $\tilde{y}_{k-1}$.

$$\hat{y}_k = \hat{y}_{k-1} - 0.1v_k = \hat{y}_{k-1} - 0.025(0.3 - \hat{y}_{k-1}), \tag{9}$$

C. Guidelines for Conducting CPS Cybersecurity Experiments

To the best of our knowledge, there are no guidelines for designing and performing CPS security experiments.

For the purpose of this study, we adapted the guidelines for case study research in software engineering, suggested by Runeson et al. [6]. Four main phases of experiments can be defined:

1) *Planning and Design*. During this phase, the system, attacks, detection techniques, variables, etc. are chosen, and the goals, research questions, and experimental design are defined;
2) *Preparation for Data Collection* – procedures for data collection are defined, and the necessary historical data is collected (such data help to understand normal system behavior and adjust detection methods' settings accordingly in order to improve detection accuracy);
3) *Data Collection* – experiment is executed and experimental data is collected;
4) *Data Analysis and Reporting* – the data is analyzed and the results are reported. The results should include responses to research questions, identified in the planning phase. Furthermore, the threats to validity of results have to be discussed.

The following sections illustrate the use of these guidelines in our experimental study. Experiment Setup Section (Section IV) described experiment phases 1 and 2, while Experiments and Results Section (Section V) - phases 3 and 4.

IV. Experimental Setup

A. System Implementation

Experiments were performed in two settings: simulation implemented in Python and conducted on MIT Soar Platform [24], and the actual robots. A State Machine (SM) [25] framework was used for modeling and the robot behavior as well as the pre-programmed attacks. The following three steps describe cyber-attack modeling on the robot using SM:

- Each elementary activity is represented as a primitive SM expressing a predicate for accepting an input. The predicate is first checked with respect to the specification and then with respect to the implementation.
- An operation is modeled as a series of primitive SMs.
- Operations are cascaded to model the attacks and detections.

Figure 4 shows a simulation framework of combinations of primitive SMs that model the robot as a discrete-time system with attacks. The framework is a result of cascading many state machines. The data from the robot's sensors is the input to the system. The *Attacks SM* contains attack models (injection, scaling, and stealthy attacks). The triangle in the figure denotes a *Gain SM* with a gain of 1. *Attacks SM* and *Gain SM* are composed in parallel to form a new SM, which is then cascaded with the next SM, i.e., *Adder SM*. The *Adder SM* decides the point of attack and attack duration. *Detections SM* contains various detection models. *Response SM* includes various response mechanisms, which are activated when attacks are detected (experiment EXP2 included the response mechanism). Finally, *Controller SM* computes the required speed of the robot to control its movement (see Figure 4).

In addition to pre-programmed attacks, actual attacks were also implemented in experiment EXP1. These man-in-the-middle attacks disrupted the wireless communication between the robot and its controller. Thus, instead of *Attacks SM*, the attack program was designed to specifically target certain packet fragments in the TCP communication between the robot and the controller. After identifying the target fragment, the program injected a Server Information Packet (SIP) fragment that matched the target fragment in the stream. Concatenating the target fragment and our rogue fragment formed a SIP that passed the checksum test of the controller and allowed us to submit false sensor measurements to the controller.

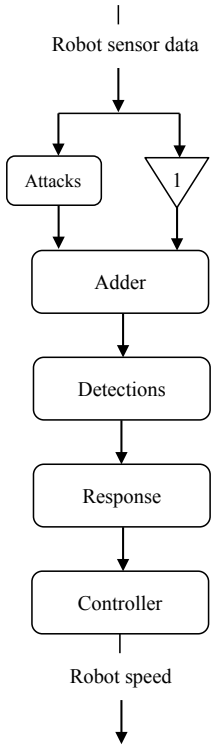

Fig. 4. Simulation framework of SM for attack modeling in Amigobot.

B. Cyber Attacks

Three types of cyber-attacks on sensor measurements were implemented: false data injection, scaling, and stealthy attacks. In the injection and stealthy attacks, sensor measurements were biased through addition, while in scaling attacks the bias was through multiplication.

In order to crash robot into a wall (achieve attacker's goal G1) (see Section III-A), an attacker can launch the following attacks:

- Injection attack - add a number greater than 0.15 to distance sensor's measurements at each time step. This attack moves the robot closer than 0.15m from the wall instead of desired 0.3m. If the attacker knows that the robot becomes unstable when it is 0.15m or less from the wall (see Section II-A), this information may be used to design the attacks and crash robot into the wall.
- Scaling attack - multiply each sensor measurement by a number. To crash robot into the wall, an attacker needs to multiply sensor measurements by 2 or a greater number. Doing so will be able the robot to reach 0.15m or shorter distance from the wall.

To force robot overshoot its destination without crashing into the wall (achieve attacker's goal G3) (see Section III-A) an attacker may add a number greater than 0 but smaller than 0.15 to sensor's measurements, or multiply sensor's measurements by a number smaller than 2.

Stealthy attacks are designed by an attacker to cause damage to the system while avoiding detection. The attacker can utilize knowledge of the system and parameters of the detection methods for this purpose. In this study, stealthy attacks were designed to avoid detection by methods M6 and M7. Three subtypes of stealthy attacks have been implemented: surge, bias, and geometric [10].

Surge attacks manipulate sensor measurements as much as possible until the CUSUM statistic reaches the threshold, at which point the falsified sensor measurements are carefully maintained to keep the CUSUM statistic exactly at the threshold. Bias attacks add a constant error to the sensor measurements at each time step, resulting in a linear growth of the CUSUM statistic up to the threshold value. Geometric attacks start slowly and increase exponentially as the attack continues.

In the following formula was used for surge attacks

$$\tilde{y}_k = max\text{-}allowable\text{-}value, \quad \text{if} \quad S_{k-1} \leq \tau \tag{10}$$

$$\tilde{y}_k = \hat{y}_k - \left|\tau + b - S_{k-1}\right|, \quad \text{if} \quad S_{k-1} > \tau \tag{11}$$

In bias attacks, $\tilde{y}_k = \hat{y}_k + c$, where $c = \tau/n + b$. In geometric attacks, $\tilde{y}_k = \hat{y}_k + \beta\alpha^{n-k}$, where $\beta = (\tau + nb)((\alpha^{-1} - 1)/(1 - \alpha^n))$. α and b are computed as described in Section II-B.

C. Variables

The following dependent variables have been used in the experiments:

- *Effectiveness*: fraction of attacks (%) detected.
- *Detection speed, or detection latency*: time required for a detection method to detect an attack measured from the moment the attack starts.
- *Detection durability*: time of detecting a continuous attack from the moment detection starts.
- *Attack effect*: impact of a successful attack on the system. The following four effects were considered:
  - *Small* - the robot is at a distance of 0.25-0.29m from the wall instead of the expected 0.3m.

  - *Medium* - the robot is at a distance of 0.195 - 0.249m from the wall.
  - *Large* - the robot is less than 0.195m from the wall.
  - *Catastrophic*: the robot crashes into the wall.

The experiments manipulated the following independent variables:

- *Attack type*: injection, scaling, and stealthy;
- *Stealthy attack sub-type*: surge, bias, and geometric;
- *Execution type*: simulation and an actual robot.
- *Attack implementation type*: pre-programmed and actual;
- *Detection method*: seven attack detection methods (M1-M7);
- *Attack timing*: attack starting time. Two different timing have been used: when the robot reached 0.8m and 0.3m from the wall;
- *Attack size*;
- *Stealthy attack duration*.

D. Experimental Goals

The overall goal of the study was to perform an extensive investigation of Amigobot's cyber security with a focus on sensor attacks. For this purpose, seven attacks and detection methods, M1-M7, have been implemented and evaluated across three experiments EXP1 - EXP3, as shown in Figure 5.

The goals of experiment EXP1 were: to test Amigobot as a possible testbed for CPS security experiments; to investigate the effect of different sensor attacks on Amigobot; and to evaluate the performance of different methods in detecting these attacks. Three Amigobots were used for this experiment in order to investigate if attack detection results are similar for different robots. Based on their performance, one of these robots was chosen for experiments EXP2 and EXP3. Simulation and three actual Amigobots were used for this experiment. Injection, as well as scaling attacks, have been implemented (see Figure 5). Pre-programmed attacks as well as actual, man-in-the-middle attacks have been executed. They disrupted the wireless communication between the robot and its controller (see Section IV-A).

The goals of the second experiment, EXP2, were: to improve the testbed based on lessons learned from the first experiment, and to further compare the performance of several attack detection methods in detecting and responding to injection attacks. Injection attacks were chosen for this experiment based on the results of EXP1: injection attacks caused greater damage to

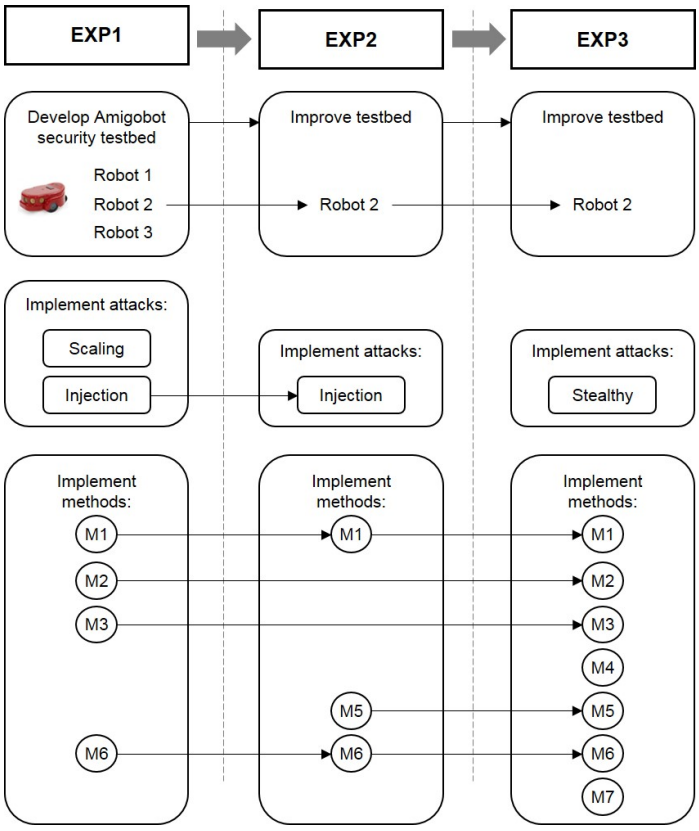

Fig. 5. Inter-relationships among experiments EXP1 - EXP3.

the robot as compared to scaling attacks (see Section V-B). The same injection attacks, as used in EXP1, were implemented in EXP2 to enable comparison of their results (see Figure 5). In addition to attack detection, a response to attacks mechanism has been implemented and evaluated in EXP2. In this experiment, an IC-based response mechanism, which replaces sensors' measurements by IC's measurements if an attack is detected, has been used.

Finally, the third experiment, referred to as EXP3, was focused on stealthy attacks (see Figure 5). The goal of the EXP3 was to further investigate the performance of anomaly detection methods, used in experiments EXP1 and EXP2, and several additional methods, in detecting stealthy attacks. The experiment was designed based on a previous study, performed by Cardenas et al. [10]. In [10], authors reported on an experiment to investigate the effectiveness of CUSUM technique [26] for detecting cyber-attacks on process control systems. The experiment was performed on a simulation model of the Tennessee-Eastman process control system [27]. In

the remainder of this paper, the experiment reported by Cardenas et al. is referred to as TEXP. The attack detection method, used in TEXP, corresponds to method M6 in our study (see Section III-B). In addition to M6, a modified version of the CUSUM method, M7, has been implemented and evaluated. The same types of stealthy attacks (surge, bias, and geometric) as in TEXP [10], were implemented in EXP3. Stealthy attacks were designed assuming the attacker is aware of CUSUM's parameters of methods M6 and M7.

E. Research Questions

1) Experiment EXP1 Research Questions:

- EXP1-RQ1. What is the effect of injection and scaling attacks on Amigobot?
- EXP1-RQ2. What methods are the most effective in detecting injection and scaling attacks?
- EXP1-RQ3. How does the detection effectiveness depend on attack size and timing?
- EXP1-RQ4. What is the difference in attack detection effectiveness, durability, and speed among three Amigobots?

2) Experiment EXP2 Research Questions:

- EXP2-RQ1: What is the effectiveness of methods M1, M5, and M6 in detecting injection attacks on sensors?
- EXP2-RQ2: What is the speed of sensor attack detection using methods M1, M5, and M6?
- EXP2-RQ3: What is the detection durability of methods M1, M5, and M6?
- EXP2-RQ4: Is the IC-based response mechanism able to prevent the robot from crashing into the wall?
- EXP2-RQ5: Is there an improvement in the effectiveness of methods M1 and M6 as compared to the results of experiment EXP1?

3) Experiment EXP3 Research Questions:

- EXP3-RQ1. What is the impact of stealthy attacks on the system when M6 is used for anomaly detection as compared to when M7 is used?
- EXP3-RQ2. How effective and timely are M6 and M7 in detecting stealthy attacks?
- EXP3-RQ3. Measured in terms of effectiveness and speed, what methods are the most suitable for complementing M6 and M7 to improve the performance in the context of stealthy attack detection?

F. Attack Detection Method Selection

Methods M1-M7, described in Section III-B, have been allocated to three experiments (EXP1, EXP2, EXP3) in the following way (see Figure 5): M1 and M6 have bee implemented in all three experiments; M2, M3, and M5 - in two experiments each; M4 and M7 - only in experiment EXP3.

The decision on which method to implement in each experiment was made based on the research questions set for each experiment, and on the results of previous experiments.

Four attack detection methods, M1, M2, M3, and M6, were implemented in EXP1 (see Figure 5). M1-M3 were simple rule-based approaches, while M6 was an estimation-based approach (see [9]). In EXP2, we kept two approaches from the first experiment (the rule-based approach M1 and the estimation-based approach M6), and added a rule-based approach that makes use additional hardware components (M5).

In EXP3, in addition to M6 and M7, five anomaly detection algorithms, M1 - M5, were implemented to determine which algorithm could be used together with M6 and M7 to improve the detection effectiveness. Thus, all seven attack detection methods (M1-M7), used in this study, have been implemented in EXP3 (see Figure 5).

G. Experiment Designs

1) Experiment EXP1 Design: The experiment manipulated several independent variables (see Section IV-C):

- Execution type. Experiment was executed in simulation, and on an actual robot in order to investigate if the detection methods' performance depended on execution environment;
- Attack starting time. We wanted to investigate whether the robot's distance from the wall at the moment attack starts influences detection effectiveness. For this purpose, two different times were used: when the robot is far from the wall (at 0.8m distance) and is moving at high speed, and when the robot has already reached its destination (0.3m) and its speed is close to 0;
- Attack type. Injection and scaling attacks were implemented;
- Attack implementation type. Pre-programmed and actual attacks were executed;
- Attack size. Several different attack sizes were implemented (see Table I).

TABLE I

DESIGN OF EXPERIMENT EXP1

	Pre-programmed attacks						**Actual attacks**
	Injection attacks				**Scaling attacks**		**Injection attacks**
	Add 0.20		**Add 0.05**		**Multiply by 2**		**Add 0.17**
Simulation/ Actual robot	**Attack start at 0.8m** A1	**Attack start at 0.3m** A2	**Attack start at 0.8m** A3	**Attack start at 0.3m** A4	**Attack start at 0.8m** A5	**Attack start at 0.3m** A6	**From beginning A7**
Simulation	5 runs	5 runs	5 runs	5 runs	5 runs	5 runs	-
Robot 1	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs
Robot 2	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs
Robot 3	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs	10 runs

Seven attack scenarios (i.e A1-A7) were defined for this experiment, as shown in Table I. Scenarios A1-A6 included pre-programmed attacks, while A7 implemented actual attacks on the communication channel between the robot and its controller.

Scenarios A1, A2, A5, A6 and A7 are aimed at achieving attacker's goal G1 - crashing the robot into the wall. A1 and A2 are false data injection attacks when 0.2 is added to sensor measurements. They differ in attack starting time: A1 starts when the robot reaches 0.8m distance from the wall, while A2 starts when the robot reaches 0.3m. A5 and A6 are scaling attacks when sensor measurements are multiplied by 2. A5 starts when the robot is at 0.8m distance from the wall, while A6 starts when the robot reaches 0.3m. A7 is an actual injection attack when 0.17 is added to sensor measurements. This attack starts as soon as the robot starts moving.

Scenarios A3 and A4 are aimed at achieving attacker goal G3 of slightly overshooting the destination. They implement injection attacks when 0.05 is added to sensor measurements to overshoot the destination by 5 centimeters. A3 starts when the robot is at 0.8m from the wall, while A4 starts when robot reaches 0.3m.

2) Experiment EXP2 Design: EXP2 manipulated the following independent variables: execution type (simulation and an actual robot), attack starting time (0.8m and 0.3m), attacks size. In addition, it implemented a response to attacks mechanism.

The design of the experiment is shown in Table II. Six attack scenarios are tested, which are implemented both in simulation (S) and in an actual robot (R):

- *S1 and R1*: small size (0.05) attack, which starts at 0.3m; no response mechanism;
- *S2 and R2*: large size (0.2) attack, which starts at 0.3m; no response mechanism;

TABLE II

DESIGN OF EXPERIMENT EXP2

	Injection attacks					
	Add 0.20				**Add 0.05**	
	Without response		**With response**		**Without response**	
Simulation/ Actual robot	**Attack start at 0.8m**	**Attack start at 0.3m**	**Attack start at 0.8m**	**Attack start at 0.3m**	**Attack start at 0.8m**	**Attack start at 0.3m**
Simulation	S4 (5 runs)	S2 (5 runs)	S6 (5 runs)	S5 (5 runs)	S3 (5 runs)	S1 (5 runs)
Robot 1	R4 (10 runs)	R2 (10 runs)	R6 (10 runs)	R5 (10 runs)	R3 (10 runs)	R1 (10 runs)

- *S3 and R3*: small size (0.05) attack, which starts at 0.8m; no response mechanism;
- *S4 and R4*: large size (0.2) attack, which starts at 0.8m; no response mechanism;
- *S5 and R5*: large size (0.2) attack, which starts at 0.3m; response mechanism is implemented;
- *S6 and R6*: large size (0.2) attack, which starts at 0.8m; response mechanism is implemented.

3) Experiment EXP3 Design: Experimental design is shown in Table III. The performance of the seven methods, M1 – M7, was evaluated in two modes: simulation and an actual robot. The three stealthy attack subtypes, namely, surge, bias, and geometric, were tailored to M6 and M7 methods to simulate the fact that the attacker knows that M6 and M7 are used for detection. In each experiment run, either M6 or M7 plus methods M1 – M5 were implemented and observed simultaneously. However, the effectiveness of each method was recorded separately.

The attack scenarios, crafted for EXP3, considered different attack starting times (as in EXP1 and EXP2) and attack durations, as we wanted to investigate if the attack duration influences the attack detection effectiveness. For this purpose, two attack durations were selected: 50 time steps and 100 time steps. For longer attack duration, a small error was added to the sensor measurements during the longer period, making it more difficult to detect the attack as compared to an attack with a shorter duration.

In total, four attack launch scenarios were implemented:

- (1) attack starts at 0.3m from the wall and lasts for 100 time steps,
- (2) attack starts at 0.3m from the wall and lasts for 50 time steps,
- (3) attack starts at 0.8m from the wall and lasts for 100 time steps;
- (4) attack starts at 0.8m from the wall and lasts for 50 time steps.

The design of the experiments is shown in Table III. Each attack scenario was implemented

TABLE III

DESIGN OF EXPERIMENT EXP3

Method	**Attack Type**[†]	**Run Type**[‡]
M1, M2, M3, M4, M5, and M6	Surge (SA1)	Simulation Robot
	Bias (BA1)	Simulation Robot
	Geometric (GA1)	Simulation Robot
M1, M2, M3, M4, M5, and M7	Surge(SA2)	Simulation Robot
	Bias (BA2)	Simulation Robot
	Geometric (GA2)	Simulation Robot

[†] The three attack types were designed assuming that the attacker is aware which of the two methods, M6 or M7, is used. [‡]Each run type corresponds to 10 executions for each of the four attack scenarios described in the text.

in simulation mode as well as in an actual robot. Anomaly detection methods M6 and M7 were implemented separately, since the surge, bias and geometric attacks were tailored for each of these methods individually (an attacker used bias and threshold parameters (b and τ) of M6 and M7 for designing each attack). Methods M1-M5 were implemented along with M6, and with M7 to investigate which of them outperformed M6 and M7, respectively. The data from 10 runs was collected for each attack.

H. Preparation for Data Collection

The same procedure was followed for all experiments. First, goals and research questions were defined. Experiment preparation, evidence collection and data analysis phases consisted of several steps (see Figure 6). To prepare for the experiment and set-up the parameters for all detection methods, historical data was collected during simulation and for each robot, under normal operation. Simulation and the actual robots were run for defined number of times to collect measurements of time steps and sensor values. Omitted from the analysis was data during the adjustment interval (10cm) at the beginning of each run due to the robot's warm-up period (see Figure 1). Then, the remaining historical data was aligned as follows. Starting with the first

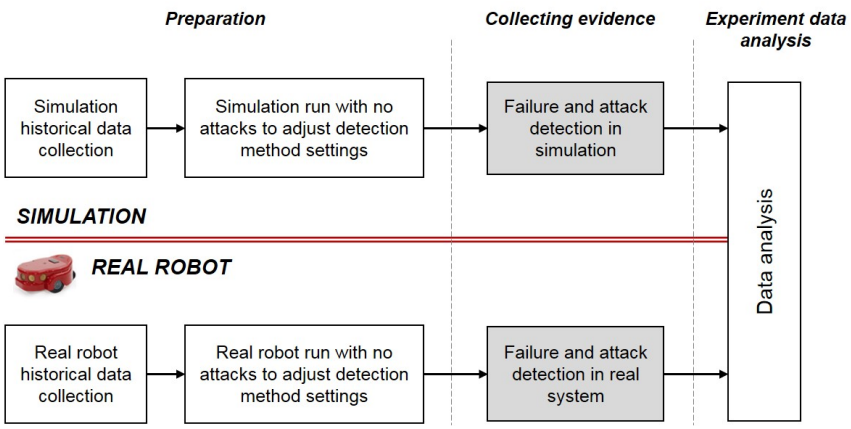

Fig. 6. Experiment phases.

time step, when the robot reached a distance smaller than 1.4m from the wall, data from 200 time steps of each run was extracted. In the simulation, the robot reached the distance of 1.4m in three time steps (0.3 seconds), while the actual robots took 10-15 steps (1-1.5 seconds) to reach this distance. The 200 time steps were chosen because this time period was long enough for the robot to reach its destination of 0.3m from the wall. After 200 steps, the robot's distance from the wall remained nearly unchanged.

Once the parameters of the detection methods (threshold values) were set, the simulation and each robot were again run 20 times without any attacks to eliminate false positives by adjusting parameters of the detection methods. Then, the system was ready for collecting and analyzing the experimental data.

In experiment EXP1, in order to set-up the parameters for attack detection methods, the simulation was run 20 times, while each robot was run 50 times to collect measurements of time steps and sensor values. Each run lasted 1 minute.

For methods M1 and M3, we took average distance and standard deviation data of 200 time steps (20s) of all runs and recorded it into separate files. Separate files have been created for simulation and actual robot. In total, four data file have been created, which included: average distance values of simulation, average distance values of the actual robot, standard deviation values of simulation; standard deviation values of the actual robot. Initially, a threshold equal to three standard deviations has been set for both simulation and the actual robot.

For method M2, we took average distance traveled and standard deviation of 200 time steps

(20s) of all runs. The initial threshold was equal to three standard deviations.

For methods M3 and M6, we calculated the bias b and threshold τ values from historical data.

Once the threshold values of the detection methods M1, M2, M3, and M6 were set, the simulation and each robot were again run 20 times without any attacks, but with the detection methods implemented, in order to adjust the settings of detection methods so that they do not detect any false positives for 2 minutes (see Figure 6). In the end, we were able to run the system for 20 times, 2 minutes each time, with zero false positives.

To collect historical data for experiment EXP2, the simulation, as well as the actual robot, were run for 20 times, each run lasted 1 minute. The parameters of methods M1 and M6 were determined in the same way as in EXP1. However, newly collected historical data has been used in order to define more precise detection thresholds for these methods and improve detection effectiveness.

For IC-based detection method, M5, the threshold τ values of the difference d_k between IC's measurements IC_k and sensor's measurements $\tilde{y}_k$, have been calculated in the following way: the distance to the wall has been divided into two periods (when the robot's distance is greater or equal to 0.55m, and when it is smaller than 0.55m from the wall), and a separate threshold has been set for each period based on the maximum difference between IC_k and $\tilde{y}_k$. This division was needed due to a significant decrease in threshold size when the robot reaches 0.55m distance. For simulation, τ = 0.02 when the robot is at 0.55m or greater distance from the wall, and τ = 0.002 when the robot is closer than 0.55m to the wall. For an actual robot, τ = 0.16 when the robot is at 0.55 m or greater distance from the wall, and τ = 0.05 when the robot is closer than 0.55m to the wall.

In the third experiment, EXP3, historical data were collected in simulation mode using 10 independent runs, and 50 runs of the actual robot.

V. Experiment Execution and Results

A. Experiment Execution and Data Collection

In experiment EXP1, three 2-student teams executed the experiment on three robots and collected the evidence of detecting pre-programmed and actual attacks. Each scenario was run 5 times in simulation and 10 times in the actual robots, as shown in Table I. Each run lasted 60 seconds or shorter; if the robot was about to crash the wall, it was immediately stopped in

order to avoid damaging the robot. For actual attack scenario, A7, each run lasted 2 minutes to give the attacker more time to succeed. Finally, experimental data have been analyzed.

The followings were data collection procedures for the actual robots. The robot has been positioned at the distance of 1.5 m from the wall before each run. In the simulator, there is a button to return the robot to the starting position after each run. In an actual robot, we used a software, implemented in Python, to move the robot to a starting position. Attack scenarios have been executed for a pre-defined number of times (as in Table I). Each run of an actual robot lasted 60 seconds or shorter: in the situations, when the robot came very close to the wall, it was stopped in order to prevent damage. After each run, the data file was labelled and saved for further data analysis.

As in Table I, in total 240 system execution runs, 30 runs in the simulation mode and 210 runs of the three actual robots were executed.

The second and the third experiments were executed by one person. In experiment EXP2, 90 system runs were performed: 30 runs in simulation mode and 60 runs of the actual robot, as shown in Table II. In experiment EXP3, 480 system runs were executed using combinations of various methods against attacks as in (Table III).

B. Results of Experiment EXP1

The results of experiment EXP1 are presented in Table IV. The attack detection effectiveness results of methods M1, M2, M3, and M6 in three Amigobots were very similar, therefore only average values of three robots were included in Table IV.

1) Results for small size injection attacks (scenarios A3 and A4): Scenarios A3 and A4 are aimed at forcing robot overshoot its destination by 5 centimeters. Attacks A3 and A4 are detected in simulation only by methods M1 and M3, while in the actual robots these are never detected (except A4 detection by method M1). This is due to a larger variation of the actual robot's sensor measurements which require higher detection threshold in order to avoid false alarms. Thus, if an attack makes small changes to sensor values, these changes might not exceed the detection threshold and therefore the attacks will remain undetected.

2) Results for large injection attacks (scenarios A1, A2, and A7): A1 and A2 implement injection attacks with the goal of crashing the robot into the wall. In A1, an attack starts when the robot is at 0.8m from the wall, while in A2, an attack starts when the robot is at 0.3m. In simulation the robot reaches 0.1m from the wall but does not crash. However, the actual robots

TABLE IV

EXPERIMENT EXP1 ATTACK DETECTION EFFECTIVENESS RESULTS

Attack Scenario Average Attack Detection Effectiveness (%)								
Attack Scenario	**M1**		**M2**		**M3**		**M6**	
	Simulation	Robots	Simulation	Robots	Simulation	Robots	Simulation	Robots
A1	100	100	100	20	100	33	100	40
A2	100	100	100	67	100	7	100	100
A3	100	0	0	0	100	0	0	0
A4	100	100	0	0	100	0	0	0
A5	100	100	100	100	100	73	100	67
A6	100	100	100	100	100	67	100	100
A7	-	54	-	8	-	8	-	23

do crash into the wall in all executions (100%) of A1 and A2. This is due to the robots becoming unstable when the distance from the wall is smaller than 0.15m.

From Table IV, in A1 and A2, all detection methods (M1, M2, M3, and M6) exhibit 100% attack detection in the simulation. However, in the actual robots, only M1 has 100% attack detection effectiveness. The other methods show low effectiveness (below 50%) if the attacks start when the robot is far from the wall (scenario A1). If attacks start when the robot is close to the wall (scenario A2), the effectiveness of methods M2 and M6 is high, while that of M3 remains low (7%).

Attack scenario A7 implements actual attacks on the robots when communication between the actual robot and its controller is disrupted via false data injection. Actual attacks were successful in all robots used for this experiment. An attacker was able to disrupt the communication between the robot and controller, and inject false data in 53% of runs. However, the closest distance reached as a result of such attacks was 0.247m. If the same attack was pre-programmed, the robot could reach 0.13m and possibly crash into the wall. This is because it was difficult for an attacker to succeed in continuously sending false data to the controller in each time step. The attacker was able to send false sensor measurements only occasionally, which allowed the robots recover their position using actual sensor measurements received in-between of false data. Method M1 was the most effective in detecting actual attacks (54%) as compared to M2 (8%), M3 (8%), and M6 (23%) (see Table IV).

3) Results for large scaling attacks (scenarios A5 and A6): Scenarios A5 and A6 are aimed at crashing the robot into the wall using scaling attacks. In simulation, the robot reaches 0.17m distance from the wall, while the actual robots reach 0.15m distance in most of the runs, and crash in 30% of the runs. This shows that injection attacks are more successful for achieving goal G1 than scaling attacks. However, it is easier to detect scaling attack as compared to the injection attack. From Table IV, all methods detect attacks A5 and A6 with 100% effectiveness in simulation, and high effectiveness (67-100%) in the actual robots.

4) Response to EXP1 Research Questions: EXP1-RQ1.This question is intended to investigate the effect of injection and scaling attacks on Amigobot. Experimental results of pre-programmed sensor attacks show that scaling and injection attacks can be used to overshoot the robot's destination, while the injection attacks were more successful in crashing the robot into the wall (in 100% of runs, while scaling attacks - only in 30% of the runs). Actual injection attacks were successful only in making the robot slightly overshoot its destination, however, they did not succeed in crashing the robot due to the attacker's inability to sustain continuous attacks. Robots were able to receive correct sensor measurements in-between of attacks and recover their safe position.

EXP1-RQ2. This question is aimed at comparing the effectiveness of attack detection methods in detecting injection and scaling attacks on sensors. Method M1 was the most effective in attack detection as it exhibited 100% effectiveness in simulation and the actual robots for attack scenarios A1-A2 and A4-A6. It only failed to detect small injection attacks on the actual robots (A3). Method M2 achieved 100% detection effectiveness in detecting scaling attacks (attack scenarios A5 and A6) but it was less effective for injection attacks (0% effectiveness for attack scenarios A3 and A4, and 20% and 67% effectiveness for scenarios A1 and A2). Method M3 had 100% attack detection effectiveness only in simulation but its detection effectiveness was low in the actual robots. Finally, method M6 was effective in simulation for detecting large injection (A1-A2) and scaling (A5-A6) attacks, however, it failed to detect small injection attacks (A3-A4).

EXP1-RQ3. This question is intended to further investigate how the attack detection effectiveness depends on additional factors besides attack type (injection and scaling), such as attack size and timing. For this purpose, we implemented three different injection attack sizes (small 0.05, and large 0.17 and 0.2), and two different attack starting times (when the robot is far from the wall 0.8m, and when it is close to the wall 0.3m). In the simulation, the detection effectiveness did

not depend on attack timing. However, in the actual robots, for methods M2 and M6, it was easier to detect attacks which started when the robot was closer to the wall. Attack detection effectiveness also depends on attack size. Small injection attacks were not detected in simulation or in the actual robots by methods M2 and M6, while methods M1 and M3 managed to detect such attacks only in simulation.

Detection method M1 exhibits 100% effectiveness in simulation and in the actual robots. However, it is unable to detect small injection attacks in A3. Methods M2, M3, and M6 were unable to detect such attacks as well. Therefore additional security countermeasures should be used to detect such attacks.

EXP1-RQ4. This question is aimed at comparing the performance of three Amigobots with respect to their attack detection effectiveness, durability, and speed. The results revealed that all of them were similar in attack detection effectiveness and durability. However, their speed of detection was different. E.g. for attack scenario A1, method M1 detected attacks on the first robot in average in 9 seconds after the beginning of attack, while in the second robot - in 11 seconds, and in the third robot - in 7 seconds. This was due to the difference in variation between different runs of the robots, which influenced the size of detection threshold. Larger threshold resulted in longer time required for detection. Based on these findings, additional historical data were collected and detection thresholds were adjusted to improve detection method performance in the second experiment.

C. Results of Experiment EXP2

The results of all attack scenarios, implemented in experiment EXP2, are summarized in Table V. The following sub-sections present detailed analysis of the results.

1) Results for small size injection attacks (scenarios S1, R1, S3, and R3): In these scenarios, small size attacks, when 0.05 is added to sensor's measurements in order to move it 5cm closer to the wall as expected, are tested.

In scenarios S1 and R1, injection starts when the robot reaches 0.3m distance from the wall and continues till the end of the run. Method M6 does not detect this type of attacks. Methods M1 and M5 immediately detect such attacks with 100% effectiveness, however, they differ in detection durability: M1 detects it in average only for 7.6s, while M5 keeps detecting it till the end of the run (42.8s). The actual robot results (R1) confirm the simulation results: method M6

TABLE V

EXPERIMENT EXP2 ATTACK DETECTION EFFECTIVENESS, SPEED, AND DURABILITY RESULTS

Attack Scenario	**Effectiveness (%)**			**Speed (s)**			**Durability (s)**		
	M1	M5	M6	M1	M5	M6	M1	M5	M6
S1	100	100	0	0	0	-	7.6	till end	-
R1	100	100	0	0	0	-	5.5	till end	-
S2	100	100	100	0	0	0	12.2	till end	1.5
R2	100	100	100	0	0	0	till end	till end	0.1
S3	100	100	0	0	0	-	17.0	till end	-
R3	30	100	0	0.7	2.4	-	1.5	till end	-
S4	100	100	100	0	0	0	17.0	till end	1.7
R4	100	100	90	0.1	0	0	7.5	till end	0.1
S5	100	100	100	0	0	0	till end	till end	2.8
R5	100	100	100	0.2	0	0	till end	till end	0.1
S6	100	100	100	0	0	0	till end	till end	2.6
R6	100	100	100	0.3	0	0	till end	till end	0.1

does not detect this type of attacks, while M1 and M5 detect it immediately, however the period of detection varies: M1 detection lasts in average for 5.5s (shorter than in simulation), while M5 keeps detecting until the end of the run. Based on simulation and the actual robot results, M1 and M5 have equally high detection effectiveness, however, M5 outperforms M1 with respect to detection durability - it is able to detect attacks from their beginning till end, while M1 only for a short while (7.6s in simulation, and 5.5s in an actual system).

In scenarios S3 and R3, the robot is far from destination (0.8m from the wall) when an attack occurs. Method M6 does not detect this attack neither in simulation nor in an actual robot. M5 exhibits high detection effectiveness (100%) in both simulation and actual system, while M1's effectiveness is high in simulation (100%), however low in an actual robot (30%). This is due to greater deviation in sensor's measurements in an actual system as compared to simulation, which influences detection threshold: M1's threshold τ is defined based on the standard deviation value, which is greater in an actual robot than in simulation, thus more attacks escape undetected. Furthermore, there is a difference in average detection durability of M1 in simulation and an actual robot: M1 detects attacks for 17s in simulation, and only for 1.5s in an actual the robot. M5 detects this attack immediately in simulation, however it has low detection speed in an actual

robot due to the large detection threshold when the robot is at 0.55m or greater distance from the wall (τ is equal to 0.16). However, when the robot reaches smaller than 0.55m distance, the threshold τ is reduced to 0.05, and such attacks are immediately detected. Furthermore, M5 detection lasts until the end of a run, while M1 detection durability is shorter(17s in simulation, and 1.5s in an actual robot).

2) Results for large size injection attacks (scenarios S2, R2, S4, and R4): In these scenarios, large size attacks, when sensor's measurements are increased by 0.2m, are tested. These attacks are aimed at moving the robot out of the safe zone, and eventually making it crashes into the wall.

In S2 and R2 each attack starts when the robot reaches 0.3m distance from the wall and continues until the end of the run in simulation. In an actual robot, the run time is shorter than in simulation, because the robot is stopped when it reaches a distance of a few centimeters from the wall in order to prevent damage of the robot. In simulation (S2), all three methods immediately detect the attacks with 100% effectiveness, however the duration of detection varies. M6 has the shortest average detection duration, only 1.5s, while M1 keeps detecting such attack for 12.2s, and M5 detects it until the end of run (42.8s). The actual robot (R2) results confirm the simulation results with respect to effectiveness: all three methods detect the attacks with 100% effectiveness. M6's detection period is the shortest, only 0.1s, i.e. it detects such attacks only once at the moment attack starts. M1 and M5 keep detecting the attacks until the end of a run, when the robot is stopped in order to avoid crashing.

In S4 and R4 attacks start when the robot is at 0.8m distance from the wall. All three methods detect such attacks immediately with high detection effectiveness and detection speed both in simulation and an actual robot. However, there is a difference in methods' performance with respect to detection durability: M6 has the shortest detection durability 1.7s in simulation and 0.1s in an actual robot; M1's detection interval is longer 17s in simulation and 7.5s in an actual robot; finally, M5 keeps detecting the attacks until the end of run both in simulation and an actual robot.

3) Results or response to large size injection attacks (scenarios S5, R5, S6, and R6): In these scenarios, large injection attacks, aimed at moving the robot out of the safe zone, are implemented. A number equal to 0.2 is added to sensor's measurements. Besides attack detection, scenarios S5, R5, S6, and R6 implement a response to attacks mechanism - corrupt sensor's measurements are replaced by IC's measurements if an attack is detected by method M5. In S5

and R5, attacks start when the robot is at 0.3m distance from the wall, while in S6 and R6 - when the robot is at 0.8m distance.

The results of scenarios S5 and R5 show that the response mechanism is able to keep system in a safe state, and ensure that the robot stays at the desired distance of 0.3m from the wall. The actual robot testing results confirm simulation results: an IC-based response mechanism is able to keep the system in a safe state. All three detection methods exhibit high detection effectiveness and detection speed in both simulation and an actual robot, however, M1 and M5 outperform M6 with respect to detection durability: M6 keeps detecting an attack for a very short time (2.8s in simulation and only 0.1s in an actual robot), while M1 and M5 keep detecting attacks till the end.

In scenarios S6 and R6, where attacks start when the robot is at 0.8m distance, it is possible to prevent the robot from entering the unsafe state, and enable it to reach its destination and maintain desired distance of 0.3m by the use of a response mechanism. All three detection methods detect such attacks with high effectiveness and speed in both simulation and an actual robot. However, M6 has the lowest detection durability (2.6s in simulation, and 0.1s in an actual robot), while M1 and M5 keep detecting the attacks as long as they last.

4) Response to EXP2 Research Questions: EXP2-RQ1. This question was intended to investigate the effectiveness of methods M1, M5, and M6 in detecting injection attacks on the robots' sensors. IC- based detection method M5 was the most effective as it detected 100% of attacks in all attacks scenarios (see Table V). Method M1 was also effective for most scenarios, however, it failed to detect attacks in an actual robot, which started when the robot was still far from destination (scenario R3) due to high detection threshold. Finally, CUSUM-based method M6 was the least effective, since it was not able to detect any small-sized injection attacks scenarios (S1, R1, S3, and R3).

EXP2-RQ2. This question was used for comparing methods M1, M5, and M6 with respect to detection speed. In most of the attack scenarios all methods exhibited fast detection speed (0-1s), as shown in Table V. The IC-based method M5 performed the best as it was able to detect attacks immediately in all but one scenarios (R3). In scenario R3, a small injection attack (0.05) is implemented, which starts when the robot is at 0.8m distance. Since M5's detection threshold τ is equal to 0.16 until it reaches 0.55m distance, it is not able to detect the attack if the robot is farther than 0.55m from the wall. However, when the robot crosses 0.55m distance,

the threshold τ is reduced to 0.05, and such attacks are immediately detected. Detection speed can be improved by defining more detection thresholds for M5 (e.g. every 10cm of the robot's path instead of having only two thresholds used in this experiment: between 1.5m and 0.55m distance, and when the distance is less than 0.55m).

The detection speed of method M1 improved in experiment EXP2 as compared to EXP1: the same robot took 11 seconds in EXP1, and only 7.5 seconds in EXP2 to detect the same attack (see EXP1-RQ4 in Section V-B4).

EXP2-RQ3. This question was aimed at comparing methods M1, M5, and M6 with respect to detection durability. IC-based detection method M5 is superior in durability since it always compares attacked sensor's value against actual value, measured by another sensor, i.e IC. Historical data based detection method M1 compares sensor's readings at each point to average historical data. So if an attacker sends data to detector similar to average historical data, the attack will not be detected. In our experiment, at a certain point the robot reaches 0.3m and average historical distance is 0.3m from that moment until the end of the run. Thus, it is enough for an attacker to send 0.3m readings to a detector. Thus, if the attacker adds 0.05m to sensor's reading, then when the robot reaches 0.25m distance, the readings sent to the detector will be 0.3m (0.25+0.05), and the attack will not be detected. CUSUM-based detection method M6 looks for a change in the difference between estimated and measured sensor's value. Thus, in the moment attack happens, it is detected, since there is a big jump in sensor's value as compared to the estimated value. However, after that, sensor's value is estimated based on previous sensor's measurements (in our case, these are the false measurements), and therefore there is no longer a big difference in values between false value in the previous step and false value in the current step. Thus, detector stops seeing the attack after a short while.

EXP2-RQ4. This question was intended to investigate the effectiveness of IC-based response mechanism in preventing the robot from crashing into the wall. Response mechanism has been implemented in for attack scenarios (S5, R5, S6, and R6), where a large number (0.2) has been added to sensors' measurements in order to move the robot out of the safe zone and possibly crash it into the wall. In all these scenarios, response mechanism prevented the robot from entering the unsafe state and enabled it to reach its destination and maintain desired distance of 0.3m. Thus, the IC-based response mechanism was effective.

EXP2-RQ5: This question was aimed at investigating if there was an improvement in effective-

ness of methods M1 and M6 as compared to experiment EXP1. Both of these methods have been used in EXP1 and EXP2, however more precise detection thresholds were calculated for EXP2 as compared to EXP1. The analysis of the results showed that in experiment EXP2 effectiveness was the same or higher as compared to EXP1 (see Tables IV and V). In EXP1, method M1 did not detect small injection attacks on the actual robots, which started at 0.8m distance (attack scenario A3 in Table IV), while in EXP2 such attacks were detected with 30% effectiveness (scenario R3 in Table V). Furthermore, in EXP1, method M6 detected large injection attacks on the actual robots, which starter at 0.8m distance (attack scenario A1 in Table IV) with 40% effectiveness, while in EXP2 - with 90% effectiveness (scenario R4 in Table V). Thus, the detection effectiveness improved in the second experiment due to more precise detection thresholds.

D. Results of Experiment EXP3

Stealthy attack detection results are summarized in Figure 7, where oval shapes encircle the attacks detected by each method.

1) Impact of attacks: Surge attacks are the least damaging as they had no effect on the system when they were designed to hide from being detected by M6. However, when M7 is used, surge attacks might cause small damage by forcing the robot to slightly overshoot its destination in scenario (3) in an actual robot. Bias attacks have no effect of the attacks in simulation, and small to medium effect on an actual robot when they are designed to avoid detection by M6. However, if M7 is used for detection, an attacker might design bias attacks that could cause small to medium damage in simulation, and medium to large damage to an actual robot.

Geometric attacks are the most damaging as compared to surge and bias attacks. They do not have an effect on system only when they are designed to hide from detection by M6, and only in simulation, while they cause medium damage to an actual robot. However, when they are designed to hide from detection by M7, they cause catastrophic damage to the system both in simulation and in an actual robot.

2) Surge Attacks: None of the methods was able to detect surge attacks in attack scenarios (1) and (2), where attacks happened when the robot was already close to the wall (0.3m distance), and surge attacks caused no effect on the system (see Figure 7.a)).

Surge attacks in scenarios (3) and (4) start when the robot is still far from the wall (0.8m distance). These attacks have no effect on the system since the robot is able to recover its

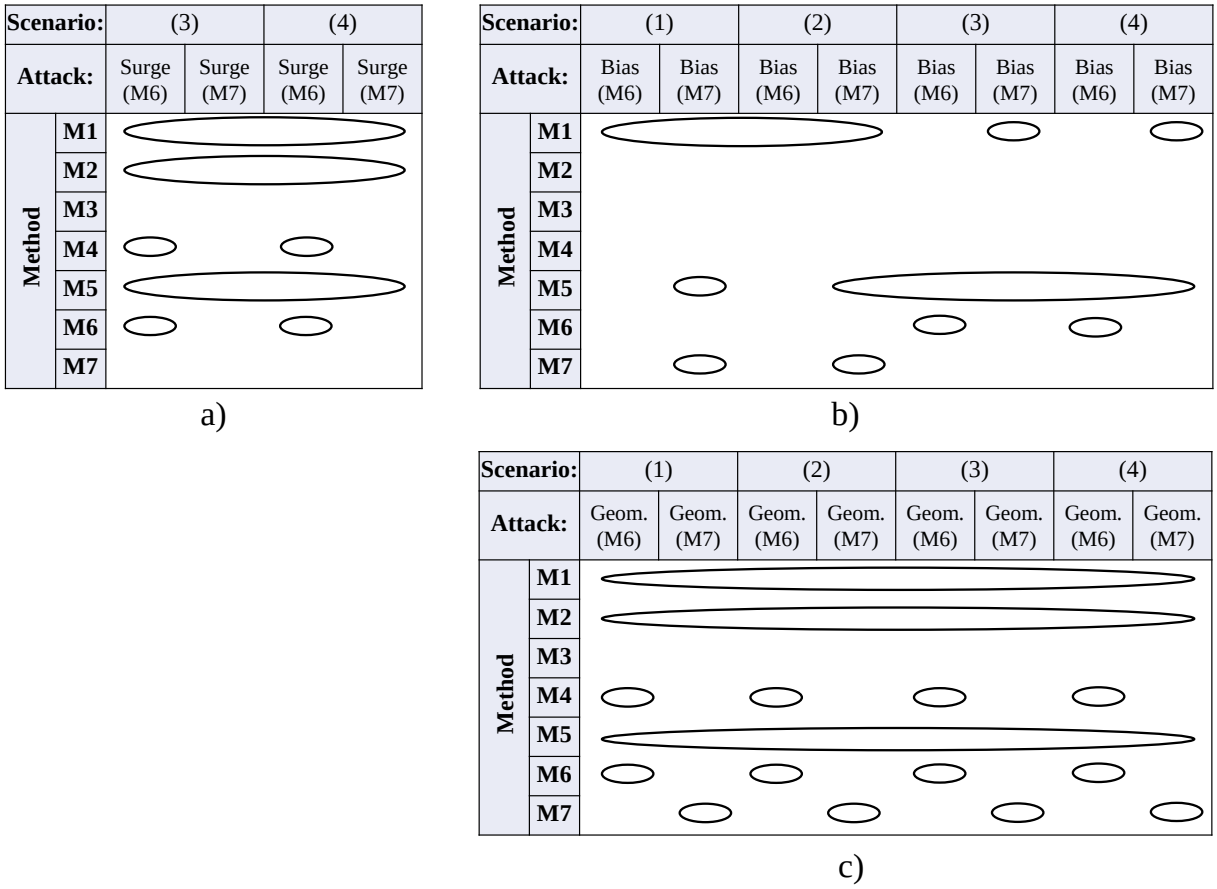

Fig. 7. Experiment EXP3 stealthy attack detection results: a) surge, b) bias, c) geometric attacks.

position after the end of attacks. However, several detection methods are able to detect these attacks during their execution. When surge attacks are designed to hide from detection by M6, method M6 is able to detect surge attacks with 100% effectiveness in simulation, and with 70-75% effectiveness in the actual robot. Detection speed is faster in simulation (0.5s) as compared to the actual robot (1-3s). Other methods that detect surge attacks are M1, M4 and M5 in simulation, and M2 and M4 in the actual robot. As M6 is able to detect surge attacks with 100% effectiveness in simulation, no additional detection method is needed when M6 is used. However, in the actual robot, M4 can be used in concert with M6, as it exhibits high effectiveness (100%) and fast detection speed (0.1s).

When surge attacks are designed to avoid detection by M7, method M7 is unable to detect surge attacks for scenarios (3) and (4). Thus, additional detection methods should be used in concert with M7. In simulation, M1, M2, and M5 exhibit 100% detection effectiveness. However,

in the actual robot, only M2 and M5 are effective (100%), while the effectiveness of M1 is only 20%. Thus, M5 is the most suitable technique to be used in concert with M7, since it has high detection effectiveness and speed as found in simulation as well as in the actual robot.

3) Bias Attacks: When bias attacks are designed to hide from detection by M6, method M6 is unable to detect bias attacks in attack scenarios (1) and (2), and has only 20% effectiveness for scenario (3) in simulation, and 22% effectiveness for scenario (4) in the actual robot (see Figure 7.b)).

For attack scenarios (1) and (2), method M1 has the highest effectiveness and speed in simulation (100%, 0.1s) and in the actual robot (90%, 0.1s). However, it is not able to detect attacks, or has low detection effectiveness, in the actual robot for scenarios (3) and (4), while M5 it has high effectiveness for these scenarios (100%). Thus, in order to detect bias attacks, both methods M1 and M5 should be used in concert with M6.

When bias attacks are designed to avoid detection by M7, method M7 is effective in detecting bias attacks only for scenarios (1) (88%) and (2) (100%) in simulation, and for scenario (2) in the actual robot (50%). Methods M1 and M5 exhibit high effectiveness (100%) and speed (0.1-0.2s) in the simulation and in the actual robot, and therefore any of them can be used in concert with M7 to improve the detection of bias attacks.

4) Geometric Attacks: If geometric attacks are designed to hide from detection by M6, method M6 is effective in detecting geometric attacks in simulation and in the actual robot, albeit at a slower rate than in other cases (see Figure 7.c)). Thus, when using M6, attacks are detected a few time steps before the end of an attack, and sometimes even after the end of an attack. M6 together with M1, M2 and M5, also leads to effective methods for detecting geometric attacks. M1 and M5 outperform M6, when used on its own, in detection speed, and hence they can be used with M6 to improve detection speed.

When geometric attacks are designed to avoid detection by M7, method M7 is effective in detecting geometric attacks only in simulation, but not in the actual robot. Methods M1, M2 and M5 have high detection effectiveness in simulation and in the actual robot. However, they differ in detection speed. For attacks scenarios (1) and (2) in the actual robot, when attack starts when the robot is at 0.3m distance, M1 has the fastest detection speed, however for scenarios (3) an (4), when the attack starts at 0.8m distance, M2 and M5 have the fastest speed. Thus, either methods M1 and M2, or M1 and M5 can be implemented to detect these attacks.

Geometric attacks cause the greatest damage to the system. When such attacks are designed

to avoid being detected by M6, they might cause a medium impact on the robot, i.e., make the robot overshoot its destination by 5-10 centimeters. However, when the system is using M7 for anomaly detection, and geometric attacks are designed using M7 parameters, the damage could be catastrophic, i.e., the robot could crash into the wall. M6 is able to detect geometric attacks only at the final steps of an attack. However, M7 is unable to detect the attacks when employed in the actual robot.

5) Response to EXP3 Research Questions: EXP3-RQ1. This question was intended to investigate the possible impact of stealthy attacks on the system, which are designed to avoid detection by CUSUM-based detection methods M6 and M7. The experimental results indicate that certain types of stealthy attacks can be extremely damaging to the system. Specifically, geometric attacks, designed to avoid detection by M7, were able to crash the robot into the wall.

EXP3-RQ2. This question was aimed at evaluating the performance of CUSUM-based detection methods, M6 and M7, in detecting stealthy attacks, designed to hide from being detected by them. M6 outperforms M7 when detecting surge attacks: M7 is unable to detect any surge attack, while M6 detects surge attacks only in the case of attack scenarios (3) and (4), i.e. when an attack starts the robot is at 0.8m from the wall. However, if an attack starts when the robot is already at its destination, it cannot be detected by M6 or M7. For geometric attacks, M6 has high detection effectiveness but slower detection speed. M7 is able to detect attacks only in simulation but not in the actual robot. This is an alarming result as geometric attacks, designed to hide from M7, are dangerous as they are undetectable while causing catastrophic damage. This indicates that M7, which relies only on system's physics model for detection, is not a suitable technique for stealthy attacks detection.

EXP3-RQ3. This question was intended to investigate the performance of methods M1-M5 in order to determine which of them is the most suitable for complementing CUSUM-based methods M6 and M7 in order to improve detection effectiveness and speed. Surge attack detection effectiveness and time to detect can be improved by complementing M6 with M4, and M7 by M5. For bias attacks, M6 should be complemented with M1 and M5. Thus, by using M1 and M5 together it is possible to assure high performance for all attack scenarios. M7 can be complemented by either M1 or M5, as both M1 and M5 have high effectiveness and speed in detecting bias attacks. Lastly, for improving the detection of geometric attacks, M6 can be complemented by either M1 or M5, and M7 by either M1 and M2, or M1 and M5, since none

of the methods has good performance for all attack scenarios.

As is evident from the discussion above, methods M1 and M5 have the best performance in detecting stealthy attacks in most of attack the scenarios, and therefore can be used to complement M6 and M7. To ensure the detection of stealthy attacks in all attack scenarios a combination of methods M1, M4, and M5 should be implemented along with M6, and M1 and M5 along with M7.

E. Combined Results of Three Experiments

This section presents a discussion of results across experiments.

EXP1 was an initial study. Three simple rule-based methods, M1-M3, and CUSUM-based method, M6, have been implemented in this experiment (see Figure 5). In EXP2, two methods from EXP1 (M1 and M6), and one additional methods M5, have been implemented, while in the third experiment, EXP3, we implemented all methods, used in experiments EXP1 and EXP2. Furthermore, two additional methods have been implemented in EXP3: rule-based method M4, and estimation-based method M7, as shown in Figure 5.

The results of EXP1 (see Section V-B) showed that method M1, which compares sensor values at each time step to the average historical values, was the most effective in detecting injection and scaling attacks on Amigobot's sensors. However, in some attack scenarios, all methods failed to detect small injection attacks, which started when the robot was still far from destination. The performance of the CUSUM-based method M6 was similar to its performance in EXP2 - it was not able to detect small injection attacks.

In EXP1 we succeeded in implementing actual attacks on all robots, which disrupted wireless communication between Amigobots and their controllers, and allowed us to submit false sensor measurements to the controllers. Although these attacks did not cause catastrophic damage and did not succeed in crashing the robot into the wall due to our inability to sustain continuously attack, they were able to make the robot overshoot its destination and get closer to the wall. Thus, if an attacker succeeds in performing continuous attacks (i.e. sending false measurements to the controller at every time step), he could crash the robot into the wall.

In EXP1 and EXP2, injection attack size influenced detection effectiveness: small-size injection attacks were more difficult to detect as compared to large-size attacks. Furthermore, attack timing influenced detection methods' performance as well: it was more difficult to detect attacks, which

started earlier in the run when the robot was still far from the wall. The most difficult attack to detect was a small-size attack which happened when the robot is still far from destination.

An IC-based response mechanism, which replaces corrupt sensor measurements by IC's measurements, was effective in EXP2 in preventing the robot from entering an unsafe state and ensuring that the robot completes its tasks in both simulation and the actual robot.

In EXP3, comparison of CUSUM-based methods with respect to their detection effectiveness and detection speed of revealed that M6 outperformed M7. Furthermore, the results of EXP3 showed that stealthy attacks could remain undetected by M6 and M7 for some attack scenarios considered while causing a catastrophic effect on the system, and therefore additional methods to complement M6 and M7, were identified and their effectiveness measured. Methods M1 and M5 had the best performance in detecting stealthy attacks in most of the attack scenarios, and therefore can be used to complement M6 and M7.

In all experiments, comparison of attack detection methods' performance in simulation and in the actual robot showed that in average the performance was similar or worse in the actual robot as compared to simulation. This is due to the amount of variation in sensor measurements between different runs in simulation and in actual robots. In simulation, the variation is very small, which allows one to define small thresholds for detection methods and to detect small attacks, which slightly exceed the threshold. However, due to a larger variation of actual sensor measurements, small attacks might not exceed the detection threshold and therefore remain undetected. Thus it is important to test detection methods not only in simulation but also on actual systems.

The experimental results showed that the characteristics of the underlying process and its model have a significant effect on the attack detection ability of various methods. Specifically, in the experiments reported here, M6 and M7 are such methods. Thus, to improve detection effectiveness, additional effort is needed to reduce the noise and in building better models.

In the absence of a reasonably accurate system model, as in the case of the Amigobot, the attack detection methods that do not depend on the system model, perform better. Such methods might use historical data and/or additional hardware for attack detection.

From the results of three experiments, we can conclude that the best performing methods for detecting corrupted sensor data in Amigobot are rule-based methods M1 and M5.

F. Threats to Validity

There are limitations and threats to validity to all empirical studies, as defined in [28], [6]. They include construct validity, internal validity, external validity, and reliability.

Construct validity refers to how well the chosen research method has captured the concerns under study, i.e. to what extent the operational measures represent what researchers have in mind and what is investigated according to the research questions. To minimize threats to construct validity, metrics were defined to help answer each research question and the same data collection and analysis procedures were followed for all attack detection methods under study in each experiment.

Internal validity is of concern when causal relationships between different factors are examined. To minimize threats to internal validity, the performance of different methods was tested in detecting attacks of different type, size and with different starting time. Furthermore, attack detection methods have been implemented on three robots in EXP1 to account for variation in mechanical properties across the robots. Historical data were collected separately for each robot in to tailor the detection method parameters and thus reduce internal validity threats.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people. CPS safety and security are of critical importance in modern society. Given that the experiments were conducted in one system, the Amigobot, the findings reported here cannot be generalized and thus the threats to external validity exist. However, the findings of this study contribute to the advancement of research in these areas as it investigates how security countermeasures can be used to enhance not only CPS security but also its safety. We are planning to implement methods M1 – M7 in other systems to improve external validity of the results of these experiments.

Reliability is concerned with to what extent the results of the study are dependent on the specific researchers. To improve the reliability of this study, strict data collection and analysis procedures were defined and used. The experiment EXP1 was performed by three student teams on different robots. Each team followed the same procedure and used the same artefacts. The other researchers can use the same experimental procedures and artefacts as in our study, but they would have to collect historical data of their Amigobot in order to adjust detection methods' parameters.

VI. Discussion

This paper proposes an approach for conducting and reporting on CPS cyber-security experiments. The approach consists of three main elements: vulnerability analysis method, attack detection methods, and guidelines for conducting cyber-security experiments.

Section III-A demonstrates that the FACT graph is a useful graphical tool for analyzing CPS vulnerabilities and identifying possible cyber-attacks. By performing vulnerability analysis before the experiment, researchers are able to identify the attackers' goals and select attacks to be considered in the experiment. For our experimental study, two safety goals were chosen, which were the most safety-critical for Amigobot, as described in Section III-A.

The FACT graph could be useful after the experiments as well for analyzing the coverage of attacks and failures by security countermeasures. The security countermeasures, tested during the experiment, could be added to the graph by attaching them to all attacks and failures they will be able to detect and/or mitigate. For example, methods M1-M7, proposed in our study, implemented statistical approaches, which detected attacks by identifying irregularities in the sensor data. Data irregularities can be caused either by sensor attacks or sensor failures. Thus, even though methods M1-M7 were primarily designed to detect attacks, they could detect failures as well.

Based on the results of experiments EXP1-EXP3, methods M1 and M5 were the most effective in detecting corrupt sensor data (see Section V-E). As they are able to detect both sensor attacks and failures, they could be attached to "Sensors' failure" nodes at Level 3 of the FACT graph instead of "Sensors' cyber attack" nodes at Level 5 (see Figure 3). This shows that sensor attacks and failures are covered by these security countermeasures and therefore no additional safety countermeasures for detecting sensor failures are needed.

The guidelines for CPS cyber-security experiments, proposed in this paper, were successfully applied in three experiments. Comprehensive description of the first experiment, EXP1, following the proposed guidelines, was very useful for designing and performing the successive experiments EXP2 and EXP3. We were able to reuse many artefacts from EXP1, such as variables, detection methods, etc., and adapt the experimental designs based on the new research questions. This allowed us to cross-compare the results of three experiments, as described in Section V-E.

We believe that our approach could help other researchers in conducting their cyber-security experiments.

VII. Conclusions

An approach for performing and reporting on CPS cyber-security experiments has been proposed and successfully evaluated in an experimental study.

In this study, three types of cyber-attacks (injection, scaling, stealthy), and seven sensor attack detection methods have been implemented. Amigobot's vulnerabilities have been analyzed using the FACT graph [5], and the attacks which could cause significant damage to the robot have been identified and empirically evaluated.

Two methods, M1 and M5 were the most effective for detecting various sensor attacks on Amigobot, and therefore could be used as an alternative to CUSUM-based methods. M1 is a simple approach, which uses historical robot's distance data for detection. Method M5, in addition to historical data, uses the measurements of intelligent checker, IC, for determining if there is an attack.

Although methods M1 and M5 had similar effectiveness, M5 outperformed M1 with respect to detection speed and durability. Furthermore, an IC-based response mechanism was effective in preventing the robot from entering the unsafe state. Thus, whenever possible, the use of ICs should be considered to improve security and safety of CPS. Currently, ICs are being implemented in SWaT testbed [3].

The results of the experiments indicate that Amigobot can be successfully used as CPS testbeds for cyber-security research. They enable evaluation of various types of attacks and anomaly detection approaches in the simulation, and the actual robot environments.

Furthermore, the proposed approach was successfully applied for describing three experiments included in this study. We will continue refining and testing it in our future experiments.

Acknowledgment

This research was supported by a start-up grant from the Singapore University of Technology and Design. It is a part of ongoing research at the iTrust, Center for Research in Cyber Security. Special thanks to Toh Jing Hui and his team for helping with real attack implementation. Thanks to Pornthip Sae-chong, Dai Gengling, Ta Susiwati, Ismam Al Hoque, Francisco Furtado, Zi Shan Lee, Noon Teo Woei Ming, Suganthi Gunasagaran, Fabien Charmet, Tan Wei Yang, Tan Ching Yi, and several others, who assisted with the experiments and data collection.

REFERENCES

[1] T. Bonaci, J. Yan, J. Herron, T. Kohno, and H. J. Chizeck, "Experimental analysis of denial-of-service attacks on teleoperated robotic systems," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ser. ICCPS '15, 2015, pp. 11–20.

[2] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, "Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 847–855, 2013.

[3] "SWaT: Secure Water Treatment Testbed," 2015, http://itrust.sutd.edu.sg/research/testbeds/.

[4] K. Junejo and D. Yau, "Data driven physical modelling for intrusion detection in cyber physical systems," in *the Singapore Cyber-Security Conference (SG-CRC)*, 2016, pp. 43–57.

[5] G. Sabaliauskaite and A. Mathur, "Countermeasures to enhance cyber-physical system security and safety," in *IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW)*. IEEE, 2014, pp. 13–18.

[6] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[7] "Amigobot," Adept Technology, Inc. 10 Columbia Drive, Amherst, NH 03031, http://www.mobilerobots.com/researchrobots/amigobot.aspx.

[8] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, J. Lu, X. Li, and N. N. Xiong, "Robust cyber-physical systems: Concept, models, and implementation," *Future Generation Computer Systems*, vol. 56, pp. 449 – 475, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15002071

[9] M. Woodard, S. Sarvestani, and A. Hurson, *A Survey of Research on Data Corruption in Cyber–Physical Critical Infrastructure Systems*. Elsevier, 2015, vol. 98, ch. 2, pp. 59–87.

[10] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: Risk assessment, detection, and response," in *ACM Symp. Inf. Comput. Commun. Security*, 2011.

[11] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly detection in wireless sensor networks," *Wireless Communications, IEEE*, vol. 15, no. 4, pp. 34–40, Aug 2008.

[12] T. Phuong, L. Hung, S. Cho, Y. Lee, and S. Lee, "An anomaly detection algorithm for detecting attacks in wireless sensor networks," *Intelligence and Sec. Informatics*, pp. 735–736, 2006.

[13] T. Peng, C. Leckie, and K. Ramamohanarao, "Information sharing for distributed intrusion detection systems," *J. Network and Comp. Apps*, vol. 30, no. 3, p. 877?99, 2007.

[14] D. Urbina, J. Giraldo, N. Tippenhauer, and A. Cardenas, "Attacking fieldbus communications in ics: Applications to the swat testbed," in *the Singapore Cyber-Security Conference (SG-CRC)*, 2016, pp. 75–89.

[15] G. Sabaliauskaite and A. P. Mathur, "Intelligent checkers to improve attack detection in cyber physical systems," in *Proceedings of the 2nd IEEE International Workshop on Cyber Security and Privacy (CSP 2013), International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2013) Beijing, PRC, in press)*, 2013.

[16] "Wibox," Spectrum CNC Technologies, 130 Industrial Way, Unit A Corona, CA 92882 USA, http://www.multi-dnc.com/products/wibox/.

[17] G. Sabaliauskaite and A. Mathur, "Design of intelligent checkers to enhance the security and safety of cyber physical systems," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, July 2014, pp. 7–12.

[18] A. J. Kornecki, N. Subramanian, and J. Zalewski, "Studying interrelationships of safety and security for software assurance in cyber-physical systems: Approach based on bayesian belief networks," in *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*. IEEE, 2013, pp. 1393–1399.

[19] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering & System Safety*, vol. 139, pp. 156–178, 2015.

[20] L. Pietre-Cambacedes and M. Bouissou, "Cross-fertilization between safety and security engineering," *Reliability Engineering & System Safety*, vol. 110, pp. 110–126, 2013.

[21] ——, "Modeling safety and security interdependencies with bdmp (boolean logic driven markov processes)," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2852–2861.

[22] "ANSI/ISA 84.00.01-2004, application of safety instrumented systems for the process industries."

[23] "ANSI/ISA-99-00-01-2007. security for industrial automation and control systems. part 1: Terminology, concepts, and models."

[24] "Soar platform," 2015, http://courses.csail.mit.edu/6.01/spring07/soar-manual/.

[25] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, 1990.

[26] E. S. Page, "Continuous inspection scheme," *Biometrika*, vol. 41 (1/2), pp. 100–115, June 1954.

[27] N. Ricker, "Model predictive control of a continuous, nonlinear, two-phase reactor," *JOURNAL OF PROCESS CONTROL*, vol. 3, pp. 10–109, 1993.

[28] C. Robson, *Real World Research, 2nd edn.* Blackwell, Malden, 2002.