Prototyping intrusion detection in an industrial cloud-native digital twin

Tärneberg, William; Skarin, Per; Gehrmann, Christian; Kihl, Maria

Published in:
2021 22nd IEEE International Conference on Industrial Technology (ICIT)

DOI:
10.1109/ICIT46573.2021.9453553

2021

Document Version:
Publisher's PDF, also known as Version of record

Prototyping intrusion detection in an industrial cloud-native digital twin

William Tärneberg*, Per Skarin†‡, Christian Gehrmann*, Maria Kihl*
*Dept. of Electrical and Information Technology & †Dept. of Automatic Control, at Lund University, Lund, Sweden
‡Ericsson Research, Lund, Sweden

***Abstract*—Digital twins are poised to play a vital role in the industry 4.0 era. A cloud-based digital twin can augment the entity that it represents. To that effect, we envision that digital twins can have embedded control systems when paired with a cyber physical system, yielding significant performance and configurability advantages. However, relegating controllers to a cloud-based digital twin exposes them to a new set of attack surfaces. Given the intricacy of such systems and the plethora of mitigating actions they can take, intrusion detection is integral to maintaining the integrity of such system. In this paper, we propose and prototype a cloud-native digital twin proof of concept for evaluating the viability of the concept. The resulting platform is evaluated for its ability to host a cyber-physical system and its potential to incorporate an intrusion detection system.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Traditional production systems and Information and Communications Technology (ICT) are converging onto what is expected to be the fourth industrial revolution, i.e. Industry 4.0. The Digital Twin paradigm is a quintessential artifact of Industry 4.0 and encompasses many of the integration and security challenges present in the Industry 4.0 domain.

In the Industry 4.0 context, a Digital Twin (DT) can be a virtual representation of anything from a component in a production plant, to a product being assembled/manufactured, to the entire production system. DTs can have different levels of abstractions and can also represent other virtual processes, including other DTs, in a hierarchical manner. More specifically, a DT maintains a state replica of the entity it represents. The stored state can be used for further analysis of the represented entity, in run-time. Further, all interactions with the represented entity go through its DT. Aggregating state presentation and analysis at a central point enables Industry 4.0 actors to observe and act, system-wide.

For scalability and expedient data-management, we also assume that DTs are realized on a cloud. In this paper, we adopt the notion that to successfully take advantage of the cloud, any application that aims to do so shall be built using Commercial off-the-shelf (COTS) cloud-native software and tools. Here, COTS encompasses anything within cloud computing, IP networks, and open interface production systems. Using COTS does not only enhance modularity and transparency but also invites non-domain specialists to build, integrate, and innovate Industry 4.0 systems.

In an industrial setting, we envision that DTs will also be able to augment the physical plant or Physcial Twin (PT), in terms of computational capacity, analytics, and security. Relegating that functionality to the DT will allow us to minimize what is at the plant and enable dynamic updates of software, practices, and policies on scalable cloud-native platforms. Arguably the most valuable beneficiaries of PT augmentation are the resident control systems. As in [8], we envision that a controller implementation that requires only a small amount of computational resources in the PT is complimented by a set of optimal or more capable controllers in the DT. The controller in the PT, the *ancillary controller*, exists as a means to keep the system functional in case the DT-side controllers are unreachable or compromised. The DT-side controllers, being it one or many, do not only have access to abundant computational capacity, they can be dynamically replaced, reconfigured, or tuned given higher level, system-wide, objectives.

Security in Industry 4.0 and in DTs is not a trivial matter [12]. We assume that a DT is more vulnerable than its PT counterpart. As with most systems, security is a complication that tend to be addressed late in the system design process. The spectrum of challenges range from device and network integrity, to software system vulnerabilities. Having relegated the plant's controller(s) to the DTs presents a new security challenge. If a control system is interfered with it can have catastrophic consequences for the plant, even fatal.

Intrusion Detection Systems (IDSs) have an important protection role in many networked systems. An IDS is one of few methods that actually has the potential to prevent zero-day attacks. Many different intrusion detection solutions have been introduced and they are often combined with evasion techniques [4]. IDSs in industrial systems are different from traditional information and communication system IDSs, in that they have to monitor activities that are frequently automated and time driven. Furthermore, such systems contain many legacy technologies, which are hard or very expensive to replace [6].

The feasibility of a cloud-native DT with a mission-critical control system and intrusion detection has not been explored in the literature. Therefore, in this paper, we implement and evaluate a cloud-native DT Proof of concept (PoC) platform. The resulting platform is deployed with a time-sensitive and

This work has been partially funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP), the ELLIIT strategic research area on IT and mobile communications, Sweden's Innovation Agency (VINNOVA) under the 5G-PERFECTA Celtic Next project, the Swedish Foundation for Strategic Research under the SEC4FACTORY project.

mission-critical Cyber-Physical System (CPS). Controllers for the CPS are implemented in the PT and the DT. Additionally, a PoC intrusion detection and mitigation strategy is implemented in the DT platform. We proceed by evaluating the performance of the resident CPS and effectiveness of the intrusion detection and mitigation strategy by subjecting the system to a set of attacks aimed at the CPS's control system. The results show that the proposed cloud-native COTS-based DT platform is able to host a CPS and that, with simple means, an intrusion and mitigation system can successfully be accommodated.

II. Related work

IDSs have a long history and play an important role in safeguarding security-critical systems and processes. The main function in an IDS is to collect data regarding suspects and analyzing the data. Mitchell and Chen makes a survey of IDSs in cyber physical systems in general and classify them according to detection technique and audit material [7]. With respect to detection techniques, knowledge based and behavior based are the main categories. Similar, one distinguishes between host and network based auditing. Works on intrusion detection specifically for control systems exist and have been thoroughly chronicled in [1], [5], [11]. In these works, cyber attacks on networked control systems are formally modeled and a set of attack scenarios are provided. The authors of [11], evaluate the models and scenarios on a real process over a wireless link to show which attack types are the most effective. The aim of [1] is to instigate a discussion between cyber-security experts and control theorists. Further, [13] focuses on detecting subtle attacks and how they can be formally mitigated.

In a DT context, host based auditing is the natural approach and especially auditing on the digital twin side. By applying the intrusion detection on the digital twin, we avoid heavy processing on the real-time physical process, while powerful detection algorithms can run on high performance computers in the digital domain. Both knowledge based and behavior based detection can be applied in a DT system. In this paper we adopt the behavior detection rule approach as suggested in [6]. This is a natural choice, since we work with a DT model built upon state analysis and state replication as we further discuss in Section III.

III. System model

In this section, we detail a general adaptation of the system model presented in [3]. The basic system model, notations, and network assumptions are from [3]. In addition to that work, we incorporate real-time feedback control as part of it's functionality. Figure 1 provides a schematic overview. A specific implementation of this model is detailed in Section V:

The system has three fundamental entities: a *Digital Twin (DT)*, a *Physcial Twin (PT)*, and a *public network*, depicted in Figure 1. The system works in discrete time k at frequency f. The components that make up those entities are as follows. The CPS under control is denoted P and referred to as the *plant*. The plant outputs its state, in message i_k, at each k.

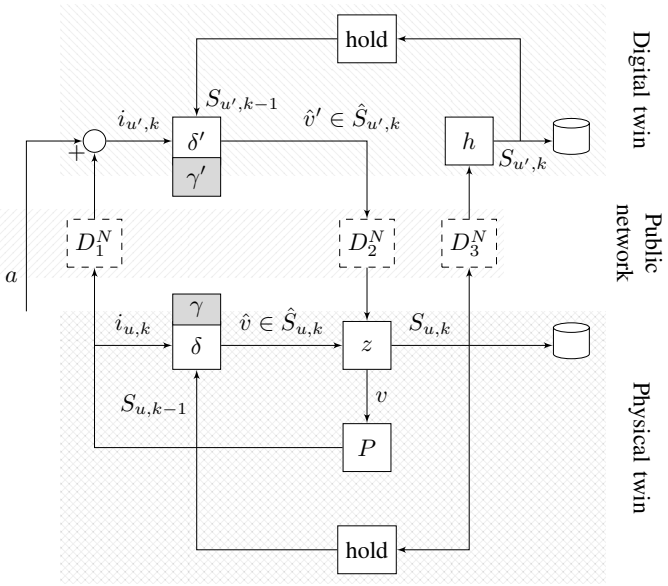

Fig. 1: Block diagram of the proposed DT system model.

Further, the plant is manipulated by a control signal v at each k. δ and δ' are the processes where the perceived state from $k-1$ meets the inputs $i_{u,k}$ and $i_{u',k}$, at k, for the physical and digital twin, respectively. δ and δ' observe and act on the PT and the DT, respectively. Consequently, δ and δ' each produce a perceived system state $\hat{S}_{u,k}$ and $\hat{S}_{u',k}$ for k, for the physical and digital twin, respectively. At process z, $\hat{S}_{u,k}$ and $\hat{S}_{u',k}$ are arbitrated, to produce the state of the physical plant at k, $S_{u,k}$. $S_{u,k}$ is sent to process h in the digital twin. The function of h is applied to $S_{u,k}$ to form $S_{u',k}$, the state of the system, at k, from the point of view of the digital twin. Further, $S_{u,k}$ and $S_{u',k}$ are recorded in Databases (DBs).

The messages $i_{u,k}$, $\hat{S}_{u',k}$, and $S_{u,k}$, between the PT and the DT are communicated over a public network and are therefore subject to a delay. The delays incurred by the network are denoted D_1^N, D_2^N, and D_3^N. The aggregate delay from the time the plant produces an output i_k to the time $\hat{S}_{u',k}$ reaches z, for k, is denoted D_z^{DT}. Similarly, the aggregate delay from the time the plant produces an output to the time it receives a control signal v, for k, is denoted D_P^{DT}.

Embedded into the system is a set of feedback controllers. The DT and the PT each have a controller, referred to as the *ancillary controllers* and *networked controllers*, respectively. The controllers, denoted γ and γ', are in processes δ and δ', respectively. Their respective control signals are denoted $\hat{v}$ and $\hat{v}'$. $\hat{v}$ and $\hat{v}'$ are tuples of arbitrary size and are specific for the controller implementation and plant. Further, $\hat{v}$ and $\hat{v}'$ are communicated in messages $\hat{S}_{u,k}$ and $\hat{S}_{u',k}$, respectively. At process z, $\hat{v}$ and $\hat{v}'$ are arbitrated to produce v.

With two controllers, γ can be augmented by γ'. In the case the DT is unreachable or intruded upon, the system falls back on γ. Another example of this controller structure can be found in [10].

A feedback controller is a real-time system that relies on

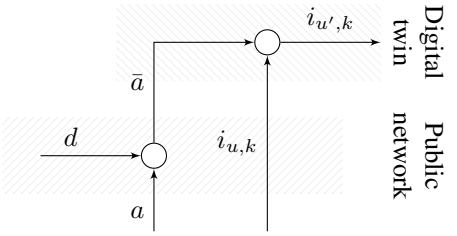

Fig. 2: Attack model for a.

timely feedback from the plant to maintain stability. To achieve stability, such systems are typically designed to operate at a near constant frequency. The controllers operate at frequencies f_γ and $f_{\gamma'}$, respectively. The operating frequency is a consequence of the properties of the controller and the plant. Typically $f_\gamma = f_{\gamma'} = f$. Consequently, to incorporate a feedback controller into the presented system model, it needs to execute at a rate dictated by the controller frequency f_γ. Analogously, $D_P^{DT} \leq \frac{1}{f_\gamma}$.

The DT receives additional inputs from the world at large through the message a. a is an auxiliary input to the DT. a can for example be a high precision measurement of the plant or an external input from another system. a is communicated over the public network. a is additive to $i_{u,k}$ and forms $i_{u',k}$.

IV. Threat Model

In this section, a threat model based on the system model described in Section III is presented. In this system, the public network, DT, and PT are considered secure, except the peripheral input a.

We use a threat model were we consider outsider and insider attacks. In previous works on DT-based security, threats on both the plant and external entities and interfaces have been considered. Meanwhile, the DT has been considered to be trusted. For instance, in [2], a threat model with reliable input and output signals is assumed to be available as well as a secure execution environment for the digital twin, while the DT itself was considered potentially compromised. In [3] a broader threat model is used. According to that model, only the state reading from the PT and the security of the digital twin can be guaranteed. We essentially adopt the threat model in [3].

In our threat model, we assume the attacker is able to manipulate the peripheral input a with the intention to influence δ'. This is indeed possible due to the fact that a is a part of the input $i_{u',k}$ to δ'. Hence, by maliciously manipulating δ' (through a), the attacker can drive the DT's controller γ', $\hat{v}'$, and ultimately v, to incur unfavorable control decisions. These can be detrimental to the operational efficiency of the plant, even catastrophic.

To incorporate the attack model, we make the following additions to the model from Section III. a is subject to manipulation d. d is the representation of the actions of an attacker. The resulting manipulated signal is denoted $\bar{a}$. This addition is illustrated in Figure 2.

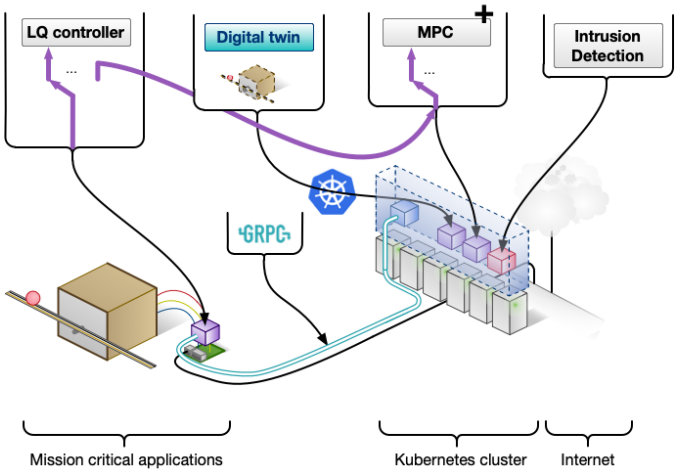

Fig. 3: A high level overview of the proposed PoC, composed of a physical plant and an edge Data Center (DC) that hosts a DT, which hosts a feedback control system as well as an intrusion detector and mitigation strategy.

An attack can be applied gradually, over a long time period, slowly driving the system to teeter into catastrophic instability. Alternatively, disturbances can be applied in bursts.

The rather simple threat model used in this paper is motivated by the fact that the peripheral input a comes from an external source and is the most likely attack target. In the future, we will extend our threat model to also cover alternative attacks, but here we limit our analysis to attacks on a.

V. Proof of Concept

In this section, we present a COTS-based DT PoC based on the Industry-4.0 test-bed presented in [9]. That test-bed was built to demonstrate the feasibility of edge computing for feedback control systems in an Industry 4.0 context. In this work, the proposed PoC builds on the components in [9] to realize the model presented in Section III and it is used to demonstrate the feasibility of a COTS-based DT and its basic security measures.

A high-level overview of the system is illustrated in Figure 3. The PoC incorporates a physical plant with an uncomplicated CPS, an edge Kubernetes (K8S) cluster, COTS-based PT and DT, and a control system that spans the production process and the edge cluster.

A. Edge K8S cluster

Proximal to the physical plant is a six-node K8S [1] cluster. K8S is an open-source container orchestration platform, often viewed as the de-facto industry standard. Additionally, K8S is extensible though a large number of open-source projects [2] ranging from security to tracing.

The six nodes of the cluster are identical, unimposing, but sufficient COTS desktop PCs, reminiscent of what one

[1] https://kubernetes.io/
[2] https://www.cncf.io/

might find in a corporate ICT infrastructure. The nodes solely host the cluster and they are connected over a solitary COTS 1000BASE-T network. Further, the cluster is shared with other tenants and their processes. The original test-bed [9] includes a set of off-site public cloud resources. Those are beyond the scope of this PoC.

The cluster has been equipped with an nginx ingress [3] and prometheus operator [4]. The nginx ingress is exposed using the K8S NodePort paradigm. Storage is realized with Rook [5].

B. Physcial Twin (PT)/Digital Twin (DT) pair

A Physcial Twin (PT)/Digital Twin (DT) pair is implemented using Python and container technology and is hosted on the K8S cluster. To reiterate, the PT is the plant-side representation of the plant and the DT is the virtual representation in the cloud. In the proposed PoC, the PT is deployed to a plant-side Raspberry Pi that can interface with the physical plant to produce $i_{u,k}$ and act on v. The DT implementation is deployed to the aforementioned K8S cluster as a *pod*.

For the PoC, the DT implementation has five primary functions: 1) communication gateway to the plant 2) plant state-store 3) feedback controller augmentation 4) input augmentation 5) intrusion detection and mitigation

All communication between the PT and the DT is over a Local Area Network (LAN) and uses a protocol defined in Protocol Buffers (Protobuf) [6] which is realized in gRPC [7]. As specified in [3] and subsequently Section III, all communication with the plant from the outside world goes through the DT. This communications includes, for example, new configurations and health checks.

The state of the plant is synced every $1/f$ seconds to the DT, using the protocol specified in Section III. The state, $S_{u,k}$, includes for example 1) the position of the ball 2) the angle of the beam 3) the control signal 4) the health of the plant 5) current controller.

The PT/DT-pair is equipped with a set of feedback controllers, γ and γ', respectively. The DT controller augments the local controller at the plant. The choice of controller can be made dynamically at the plant in a manner presented in [10] or be set externally through the communication gateway. The controllers are also implemented in Python.

C. Plant

The plant is a simple mechanical contraption, the *ball and beam* process. The ball and beam is a classic feedback control system that has a simple and clear objective. The process has a well-described model [8] and it requires timely feedback to remain stable, must be operated at reasonably high frequency, and it is a naturally constrained problem. These properties are well suited for our targeted evaluation.

[3]https://github.com/kubernetes/ingress-nginx
[4]https://github.com/coreos/prometheus-operator
[5]rook.io
[6]https://developers.google.com/protocol-buffers/
[7]https://grpc.io/

The objective of the control problem is to move a rolling ball on a horizontal beam by manipulating the beam's angle around an one-dimensional axis at the centre of the beam. The ball's targeted position on the beam is referred to as the *set-point*. The angle of the beam is adjusted by applying an input voltage to a motor that sets the angular velocity of the beam around its axis. The plant outputs the position of the ball and the angle of the beam, which is sent to a controller and used to calculate the control signal.

The plant is connected to a Raspberry Pi. The Raspberry Pi is connected to the internet and the LAN. The Raspberry Pi is constrained yet capable of running COTS software and the controller γ.

D. Feedback control system

The PoC incorporates two controllers as enabled by the system model presented in Section III. An ancillary controller (γ) in the PT (δ) and a networked controller (γ') in the DT (δ'). The plant can be successfully controlled at a rate of $20\,\mathrm{Hz}$. Therefore, both controllers operate at $20\,\mathrm{Hz}$, and dictate the system's operating frequency, $f = 20\,\mathrm{Hz}$. When γ fails to deliver $\hat{v}'$ at the targeted rate or if the DT is unresponsive, the system relies on $\hat{v}$ from γ. We refer to [10] for a full detail of the controller implementations.

γ is implemented as a Linear Quadratic Regulator (LQR). The LQR is a simple, analytically solved controller which provides a globally optimal solution but does not handle system constraints. It requires little computational capacity to produce $\hat{v}$.

γ' is implemented as a Model Predictive Controller (MPC). A MPC is an optimal controller which provides explicit specification of system constraints and must therefor also execute a numerical optimization online. The MPC runs its optimization to find a series of control signals that represent the optimal control of the plant, based on the given constraints and a prediction of the plant state. The parameter N, commonly referred to as the controller horizon, dictates the size of that series. Further, an MPC-based implementation's computational requirements grows with an increased N and are significantly greater than that of an LQR-based implementation. However, the MPC-based networked controller (γ') is intended to perform far better than the LQR-based implementation of the ancillary controller (γ).

VI. Intrusion detection and mitigation

In this section, we address the threat model presented in Section IV by applying a detection and mitigation strategy implemented in the test-bed presented in Section V. Our method adopts the rule-based intrusion detection approach from [6]. The proposed rule-based method uses the physical properties of the process in question, and detects and mitigates attacks on the speed sensor. Note that the proposed intrusion detection method is intended to prototype such a system and not to provide a general and/or best-in-class intrusion detection and mitigation strategy.

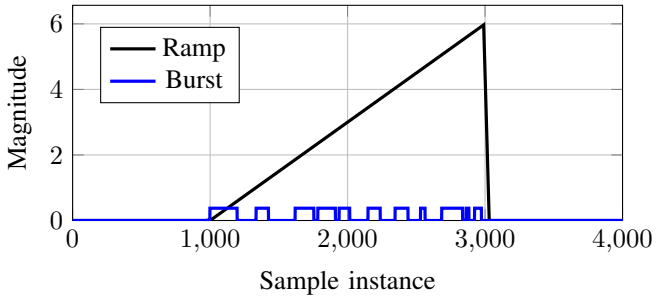

Fig. 4: Time-series representation of attacks.

The reasoning for using a rule-based method is as follows. The principal target for such an attack is the control signal v through the manipulation of a to affect $\hat{v}'$ from γ'. At δ or δ', where detection is done, it is non-trivial to determine the feasibility of the $\hat{v}'$ without replicating the γ' in δ. Replicating the controller in γ' in δ would defeat the design goal of offloading and reducing the resource footprint of the PT. Further, comparing $\hat{v}$ and $\hat{v}'$, when they are based on different controller implementations is also non-trivial. Therefore, and due to the simple nature of the plant, our intrusion detection method relies on determining the feasibility of $\hat{v}'$ given the current state of the plant.

In detail, the detection works as follows. The DT estimates the speed of the ball by observing the speed from the past observations. An estimation of the ball's acceleration is derived from the angle of the current angle of the beam, Equation (1).

$$\Delta_v = \Delta_t \cdot g\frac{5}{7}\sin(\theta) \tag{1}$$

Where g is the gravitational constant, and θ is the current angle of the beam.

The absolute difference in speed between two samples and the change in speed, is temporally averaged using an exponential moving average. When the temporal average exceeds a threshold thd_{intr}, the system is assumed to be intruded upon. A reasonable component for thd_{intr} is expected maximum difference in speed measurements from the two sensors. To mitigate the intrusion, when the system is deemed to be intruded upon, the system switches to γ.

The intrusion detection and mitigation implementation choices for the PoC are as follows. The detection is done in δ' at each k, whenever $i_{u',k}$ is received. If an intrusion is detected, that is incorporated into the proposed state $\hat{S}_{u',k}$ which is securely received by z in the PT. In the final state, $S_{u,k}$, intrusion detection outcome is computed by a logical OR operation on the individual intrusion detection outcome of the DT and the PT, in z. If an intrusion is detected, and thus the control signal from γ' has deemed unsafe, z will fall back on the control signal from a local ancillary controller γ, using the secure inputs $i_{u,k}$ or $i_{u',k}$.

VII. Experiments

In this section, a set of experiments that evaluate the validity of the COTS-based DT and the effectiveness of the proposed intrusion detection and mitigation method are detailed.

An attack and its repercussions are effectively studied in time. Therefore the duration of the experiments is kept short, at 4000 sample instances ($200\,\text{seconds}$ when $f = 20\,\text{Hz}$).

The plant and its accompanied controllers run throughout each experiment. To make detection more challenging, the plant's set-point is altered between $-0.35\,\text{meters}$ to $0.35\,\text{meters}$ every 140 samples. At that rate, under normal circumstances, both controllers γ and γ' will be able to successfully move and stabilize the ball at the set-point. Additionally, given γ and γ', that rate of set-point change allows for some disturbance tolerance.

A. Validity of COTS-based DT

Other than basic functionality, the primary metric for the validity of the proposed COTS-based DT is that it is able to execute timely. To qualify as *timely*, the end-to-end latency (D_P^{DT}), shall be within one sample instance. At $f = 20\,\text{Hz}$, one sample instance is $D_P^{DT} \leq 50\,\text{ms}$. Consequently, D_P^{DT} was measured in the system, the Round-Trip Time (RTT). To improve the contrast of the results, the execution time as well as the infrastructure primitive latency is provided.

B. Effectiveness of intrusion detection and mitigation method

The effectiveness of the proposed intrusion detection and mitigation method from Section VI, is primarily determined by its ability to timely detect the attack and mitigate its effects. Here, *timely* means that an attack shall be detected and mitigated before the plant reaches a catastrophic state. Secondarily, the proposed method shall be able to do so over a set of attack types. Thirdly, as γ' provides the preferred control signal $\hat{v}'$, the proposed method shall be able to detect when the attack is over and consequently revert back to γ' from γ.

As stated in Section IV, the targeted attack is a malicious manipulation of a as a means to influence γ', to produce control decisions $\hat{v}'$ that will lead to degraded performance and/or catastrophic failure of the plant. Here, a is an improved speed measurement for the benefit of the MPC in γ'.

Assuming no access to the rest of the system nor any knowledge of the system's model, it is assumed that an attacker does not have sufficient context to produce reasonable speed values. Therefore, the attacker applies a Gaussian noise to a. Based on this notion, two types of attacks are presented below.

1) Attack: We subject the intrusion detection and mitigation method to two types of attacks. In both cases, Gaussian noise is applied to a. An attack is conducted between sample instance $k = 1000$ and $k = 3000$. For simplicity, k^a denotes a sample instance in space of the attack. Both attacks are designed to lead to a catastrophic outcome. Note that the parameters of the attacks have been tuned to, on average, produce a catastrophic failure in the last 3rd of the attack window.

Ramp A Gaussian noise is applied to a, increasing with time. Formally, $\mathcal{N}(0, \sigma(k_i^a)^2)$ where sigma is increased linearly with time, $\sigma(k_i^a) = A \cdot (k_i^a - k_0^a)$, where $A = 0.005$. Thus, $\bar{a}(k_i^a) = a(k_i^a) + \mathcal{N}(0, \sigma(k_i^a)^2)$.

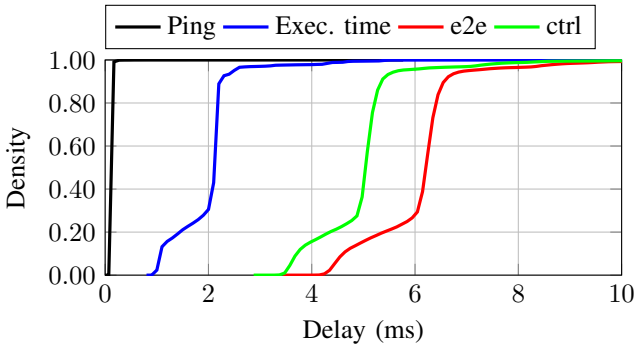

Fig. 5: Round trip time for different stages of the DT

Burst A time-invariant Gaussian noise is applied to a in ON/OFF bursts. An ON period is followed by an OFF-period. The duration of such a period is sampled from a uniform distribution, $\mathcal{U}(b,c)$, where $b=10, c=200$. An attack consists of as many bursts that can fit into the designated attack time-frame. Consequently, during an ON-period, a is manipulated as $\bar{a}(k_i^a) = a(k_i^a) + \mathcal{N}(0,\sigma^2)$.

We rely in the metric Relative Accumulated Error (RAE) to measure the impact of the attack on the performance of the controller. RAE is a measure of the controllers ability to reach the requested set-point. p_k is the state (scalar) at sample k and r_k is the set-point at sample k.

$$\frac{1}{T}\sum_{k=0}^{T}|p_k - r_k| \tag{2}$$

VIII. Results

In this section, the results from the experiments detailed in Section VII are presented.

A. Validity of COTS-based DT

1) Latency: As a basic requirement, the system's response time shall be able to accommodate the controllers' sampling rate, (γ and γ'). Both controllers work at 20 Hz, thus $f = 20\,\text{Hz}$. The desired outcome is thus $D_P^{DT} \leq 50\,\text{ms}$.

The line furthest to the left in Figure 5 shows the ICMP echo response time between the plant and the K8S cluster. In the absence of a cloud and a DT-platform, this infrastructure RTT primitive is $0.13\,\text{ms}$, which is significantly below the $50\,\text{ms}$ target.

The execution time of the resident controller γ' in δ' is represented by the line second to the left Figure 5. At a mean delay of $1.97\,\text{ms}$, with a large margin, the execution time of γ' can be accommodated for.

The line second to the right in Figure 5 shows D_P^{DT}. At this point, having transgressed the K8S platform, at a mean of $4.94\,\text{ms}$, the delay is significantly greater than the sum of the execution time of γ' and the infrastructure RTT primitive. Since 99.99% of messages experience an average below $8\,\text{ms}$ delay, the proposed platform is capable of hosting the ball and beam process. Further, the mean total end-to-end latency, from the time the plant outputs a signal to the instance $S_{u,k}$ arrives at h and $S_{u',k}$ is produced, is $6.07\,\text{ms}$.

Noise	None		Ramp		Burst	
IDS	No	No	No	Yes	No	Yes
Controller	γ	γ'	γ'	γ' & γ	γ'	γ' & γ
RAE	0.06	0.05	0.26	0.05	0.21	0.05

TABLE I: RAE for all scenarios.

2) Rudimentary controller performance: As a controller performance baseline, the first 1000 sampling instances in each plot in Figure 6 show the plant's response to control signal $\hat{v}'$ from γ'. The figure shows the position of the ball and the set-point in the span of an experiment. During these periods, the system is not subject to an attack. Therefore, γ' is the active controller. Evidently, the controller is able to maintain stability as it is able to keep the ball on the beam and successfully and timely move the ball to a fleeting set-point. This outcome will act as our controller performance baseline. It can therefore be concluded that the proposed DT platform and the resulting PoC are functional and we can move on to evaluate the intrusion and mitigation method

B. Effectiveness of intrusion detection and mitigation method

With a functioning PoC and a controller performance baseline we can begin to assess the effectiveness of the proposed intrusion detection and mitigation method. The upper two plots in Figure 6 show the outcome of the attacks detailed in Section VII-B1, without any detection nor mitigation. The grey-shaded area demarks the time-period under which the system is under attack. As presented in section VII-B1, the general metric RAE quantifies the performance of the controller over a time window. Table I holds the RAE-value for each experiment. The two first columns shall be seen as performance baselines. In particular, with γ', no noise, and no intrusion mitigation and detection (IDS) case is the expected best performer.

The upper left most plot shows the outcome from a ramp attack. At $k = 2381$ the system succumbs to the attack and the ball falls off the beam. Further, the upper right most plot shows the outcome of a bust attack. Here, the ball falls of the beam at $k = 2800$. In either case, the performance of the controller γ' is negatively affected. From Table I, in contrast to the baseline case (RAE$= 0.05$), the RAE drops to 0.26 and 0.21, for the ramp and bust attacks, respectively. These are significant and undesirable reductions in performance. In both cases the RAE is measured up until the point where the ball falls off the beam.

The two lower plots show the attack outcomes when intrusion detection and mitigation is enabled. The blue-dashed area demarks when the intrusion detector determines that the system is under attack. Consequently, in those areas, $\hat{v}$ from γ is applied to the plant.

The proposed method is able to detect the attack in 29 and 8 sample instances, for the ramp and burst attacks, respectively. Further, the method detects that the attack has subsided after 216 and 122 sample instances, for the ramp and burst attacks, respectively. In either case, the system is successfully able to

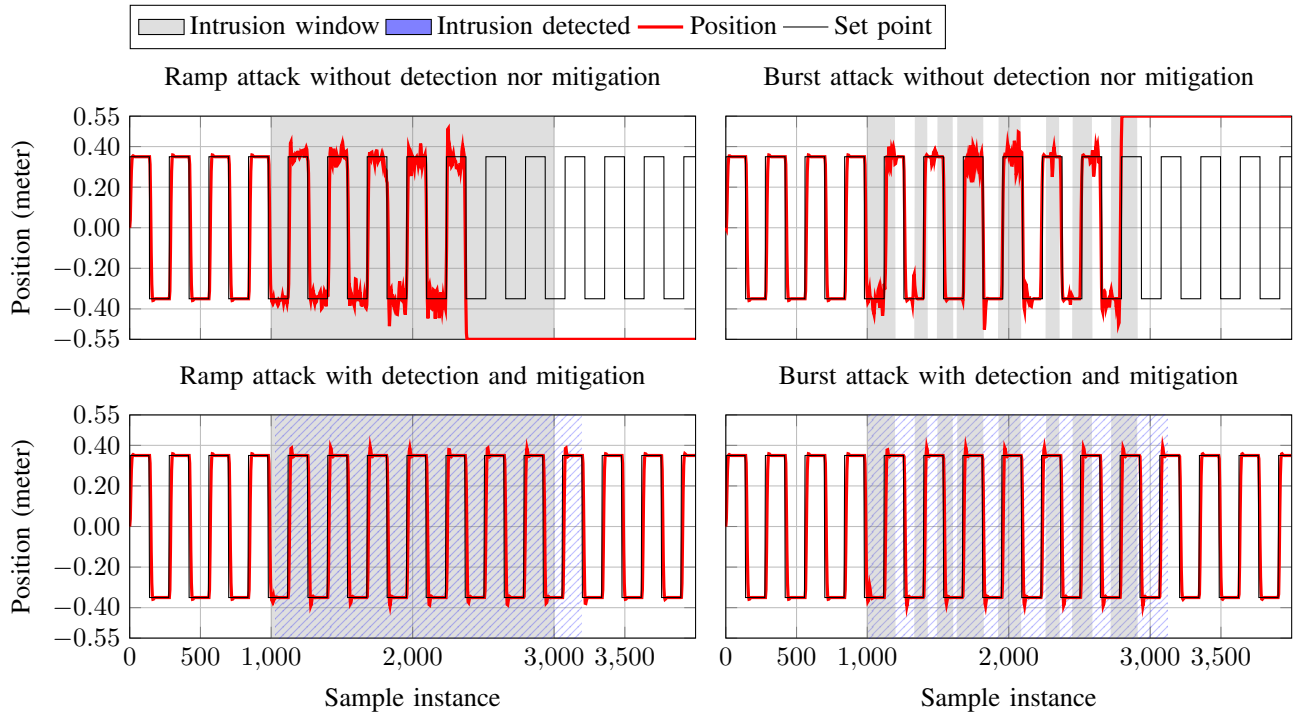

Fig. 6: Result of experiments: The effects of intrusion detection and the success of detection and mitigation.

quickly determine when the system is under attack. It is worth noting that a lag at the tail-end is tolerable, and in some cases perhaps even desirable.

Note the reduction in the ball's erratic movements when the attack is detected and mitigated. Although γ' is the desired controller, the system is able to quickly detect the attack and switch over to γ. Additionally, from movement of the ball in the two lower plots in Figure 6, it is analogous when the system reverts back to γ' once the attack has subsided. Consequently, one can conclude that the detection and the resulting mitigation is timely. In fact, as seen in Table I, there is no drop in RAE, in either case.

IX. Conclusions

In this paper, the validity of a COTS cloud-native DT with an embedded control system and intrusion detection for the industry 4.0 era was investigated. The evaluation was done on a PoC that has a CPS, a Kubernetes cluster, and a COTS-based cloud-native PT/DT pair. A demonstrative intrusion detection method that exploits the state of the plant was implemented in that system. The evaluation shows that such a system can feasibly accommodate a CPS. Further, experimentation showed that the intrusion detection and mitigation can be accommodated in the architecture and with the performance of the distributed cloud infrastructure. Additionally, the system was able to effectively mitigate the attack and maintain the integrity of the CPS.

References

[1] Alvaro A Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Symposium on information, computer and communications security*. ACM, 2011.

[2] Matthias Eckhart and Andreas Ekelhart. Towards security-aware virtual environments for digital twins. In *Workshop on Cyber-Physical System Security*. ACM, 2018.

[3] Cristian Gehrmann and Martin Gunnarsson. A digital twin based industrial automation and control system security architecture. *IEEE Transactions on Industrial Informatics*, 2020.

[4] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Oxford University, Journal of Cybersecurity*, 2019.

[5] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. In *IFAC Symposium on Information Control Problems in Manufacturing*. Elsevier, 2018.

[6] Robert Mitchell and Ing-Ray Chen. Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. *IEEE Transactions on Systems, Man, and Cybernetics*, 2014.

[7] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computer Survey*, 2014.

[8] Per Skarin, Johan Eker, Maria Kihl, and Karl-Erik Årzén. Cloud-assisted model predictive control. In *International Conference on Edge Computing*. IEEE, 2019.

[9] Per Skarin, William Tärneberg, Karl-Erik Årzén, and Maria Kihl. Towards mission-critical control at the edge and over 5G. In *International Conference on Edge Computing*. IEEE, 2018.

[10] Per Skarin, Tärneberg William, Karl-Erik Årzén, and Maria Kihl. Control-over-the-cloud: A performance study for cloud-native, critical control systems. In *Accepted to UCC*. IEEE, 2020.

[11] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. Attack models and scenarios for networked control systems. In *Int. conf. on High Confidence Networked Systems*. ACM, 2012.

[12] Lane Thames and Dirk Schaefer. *Cybersecurity for industry 4.0*. Springer, 2017.

[13] David I Urbina, Jairo A Giraldo, Alvaro A Cárdenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Conference on Computer and Communications Security*. ACM, 2016.