Saarland University
Department of Computer Science

Security Aspects of Anomaly Detection for Cyber-Physical Systems

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

von
Alessandro Erba

Saarbrücken, 2023

Tag des Kolloquiums:	29. April 2024
Dekan:	Prof. Dr. Roland Speicher

Prüfungsausschuss:

Vorsitzender:	Prof. Jan Reineke
Berichterstattende:	Dr. Nils Ole Tippenhauer
	Prof. Martina Maggio
	Prof. Mario Fritz
	Prof. Emil Lupu
Akademischer Mitarbeiter:	Dr. Zheng Li

Zusammenfassung

Cyber-physische Systeme (CPS) führen autonom Aufgaben in der physischen Umgebung aus. CPS nutzen dabei Rechenressourcen, Sensoren, Aktuatoren und Kommunikationsprotokolle. Beispiele solcher Systeme sind Industriesteuerungssysteme (ICS) und unbemannte Luftfahrzeuge (UAV). In der Vergangenheit kam es bereits zu Angriffen auf solche Systeme, die Menschen gefährdeten und Umweltverschmutzung sowie wirtschaftliche Verluste verursachten. Zur Reduzierung des Risikos von Angriffen auf CPS wurde Anomalieerkennung (AD) vorgeschlagen.

In dieser Arbeit erforschen wir Sicherheitsaspekte von CPS und der Anomalieerkennung. Zunächst untersuchen wir die Sicherheit moderner ICS-Protokolle und zeigen, dass der Einsatz sicherer Systeme oft unmöglich ist. Zweitens untersuchen wir die Sicherheit von UAVs und schlagen Sensor Deprivation Attacks vor, einen neuartigen Angriffsvektor, der eine feindliche Steuerung des Opferfahrzeugs ermöglicht. Drittens untersuchen wir die Sicherheitsaspekte der Anomalieerkennung für CPS bei gezielten Verschleierungsangriffen zur Umgehung von Klassifikatoren. Wir bewerten die Robustheit von prozessbasierten AD-Systeme aus der Literatur gegen Angriffe, die darauf abzielen, Prozessanomalien zu verschleiern. Wir schlagen drei Ansätze vor, um die Sicherheit von AD-Systemen zu bewerten, indem wir Angreiferbeschränkungen, Erkennungseigenschaften und minimale Störungsgrenzen erforschen. Unsere vorgeschlagenen Ansätze ermöglichen eine Sicherheitsanalyse von AD-Systemen.

Abstract

Cyber-Physical Systems (CPS) autonomously accomplish tasks in the physical environment. CPS employ computational resources, sensor, actuators, and communication protocols. Examples of such systems are Industrial Control Systems (ICS) and Unmanned Aerial Vehicles (UAV). Their security is of primary importance in our society. Attacks on such systems occurred in the past, harming humans, and causing environmental pollution, and economic losses. To mitigate the risk of attacks on CPS, anomaly detection (AD) techniques have been proposed.

In this thesis, we systematically assess the security of CPS and the security aspects of anomaly detection. First, we explore the security of modern ICS protocols and show how deploying secure systems is often impossible. Second, we investigate the security of UAVs, we propose Sensor Deprivation Attacks, a novel attack vector that enables adversarial control of the victim vehicle. The proposed attacks are stealthy from state-of-the-art UAVs AD. Third we investigate the security aspects of anomaly detection for CPS when targeted with concealment attacks for classifier evasion. We evaluate the robustness of process-based AD proposed in the literature against attacks that aim to conceal process anomalies. We propose three frameworks to assess the security of AD by exploring attacker constraints, detection properties, and minimal perturbation boundaries. Our proposed frameworks enable the first systematic security analysis of CPS anomaly detectors.

Background of this Dissertation

This dissertation is based on the papers mentioned in the following. I contributed to all documents as the main author. The contributions are sorted in chronological order.

Chapter 6 is based on the research paper [P1], which was inspired by the preliminary work on my master's thesis [58]. In addition to the content of the thesis [58], the paper introduces the novel concept of constraints for the concealment attacker, and the constraints formalization. Based on the definition of the constraints, the paper explores the impact of the constraints on the evasion capability of the attacker. The definition of the constraints, the evaluation of such, and the comparison with the baseline replay attack constitute the novelty presented in Chapter 6. The initial idea of exploring constraints for the attacker came from Alessandro Erba. Alessandro Erba was responsible for the implementation and evaluation of the proposed constraints. Feedback about the design and evaluation of the attacks came from all the co-authors of the paper. All authors participated in the writing and reviewing of the paper.

Chapter 3 is based on the research paper [P2], resulting from a collaboration with Anne Müller, and Nils Ole Tippenhauer. The initial idea of exploring the security of OPC UA protocol came from Alessandro Erba and was refined by Nils Ole Tippenhauer. The design of the evaluation methodology was jointly defined by Alessandro Erba and Nils Ole Tippenhauer. The evaluation was jointly conducted by Alessandro Erba and Anne Müller. Feedback about the design and evaluation of the attacks came from all the co-authors of the paper. All authors participated in the writing and reviewing of the paper.

Chapter 7 is based on the research paper [P3], while Chapter 8, is based on the research paper [P4]. Both papers resulted from the collaboration with Nils Ole Tippenhauer. Alessandro Erba proposed the idea of generic concealment attacks, white-box concealment attacks. Alessandro Erba was responsible for the evaluation and paper writing. Nils Ole Tippenhauer actively provided feedback on the design, and the experimental evaluation and participated in writing and reviewing the text.

Chapter 5 combines content from papers [P1, P3, P4] to provide a unified view of the approaches proposed in Chapters 6, 7, 8.

Chapter 4 is based on the research paper [P5] (which is currently in preparation for submission). This paper is resulting from a joint work with John Henry Castellanos, Sahil Sihag, Saman Zonouz, and Nils Ole Tippenhauer. The initial idea of exploring stale attacks on UAVs came from Saman Zonouz and Nils Ole Tippenhauer. The idea of exploring Reinforcement Learning approaches for adversarial control came from Alessandro Erba. The idea of implementing the sensor deprivation attacks via sensor reconfiguration came from Alessandro Erba with the proposal of the suspend attack, which was refined by John Henry Castellanos with the idea of frequency attacks. The implementation and evaluation were jointly conducted by Alessandro Erba and John Henry Castellanos, while Sahil Sihag was responsible for re-implementing the M2Mon anomaly detector from prior work, and its evaluation. Saman Zonouz and Nils Ole Tippenhauer provided feedback at the different stages of development of the work. All authors participated in the writing and reviewing of the paper.

[P1] Erba, A., Taormina, R., Galelli, S., Pogliani, M., Carminati, M., Zanero, S., and Tippenhauer, N. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Austin, USA, Dec. 2020.

[P2] Erba, A., Müller, A., and Tippenhauer, N. Security analysis of vendor implementations of the opc ua protocol for industrial control systems. In: *Proceedings of the 4th Workshop on CPS & IoT Security and Privacy (CPSIoTSec '22)*. ACM, Los Angeles, CA, USA, Nov. 2022. **Best Paper Award**.

[P3] Erba, A. and Tippenhauer, N. Assessing model-free anomaly detection in industrial control systems against generic concealment attacks. In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Austin, USA, Dec. 2022. **Distinguished Paper with Artifact Award**.

[P4] Erba, A. and Tippenhauer, N. White-box concealment attacks against anomaly detectors for cyber-physical systems. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Nature Switzerland, Cham, 2023, 111–131. Reproduced with the permission from Springer Nature. **Best Paper Award.**

[P5] Erba, A., Catellanos, J. H., Sihag, S., Zonouz, S., and Tippenhauer, N. Sensor Deprivation Attacks for Stealthy CPS Manipulation. Paper in preparation for conference submission. 2023.

Further Contributions of the Author

I also contributed to the following papers.

[S1] Turrin, F., Erba, A., Tippenhauer, N., and Conti, M. A statistical analysis framework for ICS process datasets. In: *Proceedings of the Joint Workshop on CPS&IoT Security and Privacy (CPSIOTSEC'20)*. ACM, Nov. 2020.

[S2] Walita, T., Erba, A., Castellanos, J. H., and Tippenhauer, N. Blind concealment from reconstruction-based attack detectors for industrial control systems via backdoor attacks. In: *Proceedings of the Cyber-Physical System Security Workshop (CPSS)*. 2023.

Acknowledgments

I would like to express gratitude to my advisor Dr. Nils Ole Tippenhauer. Thank you for the invaluable support during my doctoral journey. Always available to support me, to listen, and to advise. I learned a lot from you, and I will be always in debt.

To my colleagues Till, Yu-De, and Simeon. Thanks for the time together and the mutual support. It was a pleasure sharing those years with you. I will bring with me the memories of the time we shared. I also want to thank all collaborators and students, that I was lucky to work with.

To Prof. Saman Zonouz, thanks for mentoring and hosting me at Georgia Tech.

I would like to express my gratitude to the Helmholtz Center for Information Security for supporting my career development during my doctoral studies. I would like to express special gratitude to the Travel, the HR, and the PhD Coordination departments. Especially Ramona, Daniela, Peter and Heike.

To the friends in Saarbrücken: Annalisa, Benedetta, Michael, Riccardo, Lucía, Renato, Chiara, Antonio, Paolo, Stefano and Francesco. Thank you for the road we shared, and the time together. I could not ask for better company. To the friends in Atlanta: Lorenzo and Branson. Thanks for the time together, your company shaped fundamentally my experience. To the friends in Luxembourg: Alessio, Maria Cristina, Luciano and all the LUX friends. I am grateful for the experience we shared. To Biagio, thank you for always being there to support me. To John, Caro, and Elena, I will always be grateful for our friendship.

To my parents, thanks for your unconditional support and the example you are for me. To my brother, thanks for inspiring me since childhood, your example is a continuous resource. To Guia, thank you for your unconditional optimism. Carlotta, Matilde, and Riccardo, thanks for being so curious. To my grandparents, thanks for the example you are to me. To my extended family, especially Alfredo and Dina, thanks for the support over the years.

To my son Edoardo, thanks for the gift you are. To my wife Elisabetta, thanks for choosing to walk this road together. This doctoral journey would have not been the same without your everyday support.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Modern society relies on Cyber-Physical Systems (CPS) to automate processes through the interaction of computational resources, computer networks, and the physical environment [119]. For example, the water that reaches our apartments relies on cyber-physical infrastructures like water treatment plants and water distribution systems. CPS is deployed to accomplish several tasks, CPS can vary its architecture based on the application, ranging from small embedded devices (like quadcopter drones and Autonomous Vehicles) to geographically distributed infrastructures (like power grids and Industrial Control Systems). The security of CPS is a major concern in our society, as attacks against CPS infrastructures occurred in the past and can harm people, the environment, and the economy [3, 202, 204]. Securing Cyber-Physical Systems is an open problem, crossing all CPS classes.

Industrial Control Systems (ICS), for example, are designed for long lifecycles and rely on expensive components deployed in the field for decades [181]. Such systems often rely on legacy devices that communicate via insecure protocols. Software updates for such devices are also less likely as they might cause compatibility problems among heterogeneous appliances in the network. ICS architectures are application-specific and often rely on ad-hoc solutions that are unique for the considered plant (e.g., for intellectual property and industrial secrets). Autonomous Vehicles rely on relatively shorter lifecycles compared to ICS. Thousands of units of the same vehicle model are produced with the same architecture and components. Software and firmware can be regularly updated over the air as the hardware components are well-defined [86]. On the other side, such systems operate in the public environment and rely on the trustworthiness of the perceived surroundings [48].

Despite the different architectures in the CPS domain, the correct functioning of any CPS relies on the control loop (see Chapter 2). Attacks to CPS aim at destabilizing, or modifying the control loop to produce wrong control decisions or damage the CPS hardware. For example, in the Stuxnet attack [202], the attacker implanted a Programmable Logic Controller (PLC) rootkit to modify the control logic of the victim control program to change the rotation frequencies of centrifuges actuated by the PLC to break them with the ultimate goal of disrupting nuclear processes. At the same time, attackers concealed the attack by reporting the correct speed to the monitoring system.

To secure Cyber-Physical Systems, a number of solutions have been proposed. In particular, novel secure-by-design protocols for next-generation CPS [27] and anomaly detection using Machine Learning techniques to identify ongoing physical process anomalies [32]. The introduction of secure-by-design protocols guarantees the authenticity of exchanged information within the physical process and the controller, preventing message manipulations in the industrial network. Anomaly detection, on the other hand, is proposed to easily integrate into insecure legacy systems to react to anomalies occurring on the system before they damage the system functionalities. However, little is known about the security guarantees of such solutions and countermeasures.

In this thesis, we systematically assess the security of CPS by evaluating secure-by-design ICS protocols, proposing novel attacks to emerging CPS architectures, and proposing frameworks to assess the security of process-based anomaly detection. Our results show the limitations of state-of-the-art solutions to CPS challenges. We contribute

to the security of CPS by proposing frameworks and techniques to systematically verify how effective are security solutions for Cyber-Physical Systems.

Our contribution raises awareness about the security aspects of novel technologies proposed to defend CPS and shows how, if not carefully designed and integrated, they can result in a false sense of security.

1.1 Contribution and Organization

In Chapter 2, we discuss background on Cyber-Physical Systems and Adversarial Machine Learning relevant to this thesis. The thesis is then divided into two main parts. In Part I, we evaluate the security of modern ICS protocols (Chapter 3) and state-of-the-art autonomous robotic vehicles (Chapter 4). In Part II, we evaluate the security aspects of process-based anomaly detection systems for CPS (Chapter 5, Chapter 6, Chapter 7 and Chapter 8). Chapter 9, concludes the thesis by summarizing the findings. We will review related work for each chapter individually when required.

Specifically, the thesis contributions are the following:

- In Chapter 3, we systematically investigate 48 publicly available artifacts of the secure-by-design OPC UA protocol and show that 38 out of the 48 artifacts have one (or more) security issues. Consequently, relying on those products and libraries will result in vulnerable implementations of OPC UA security features. To verify our analysis, we design, implement, and demonstrate attacks in which the attacker can steal credentials exchanged between victims, eavesdrop on process information, manipulate the physical process through sensor values and actuator commands, and prevent the detection of anomalies.

- In Chapter 4, we propose *Sensor Deprivation Attacks* (SDAs) in which the update rate of sensors is reduced (or sensors are fully disabled) in a way that is transparent to the sensor data consumer, i.e., the control algorithm. We propose a novel attack methodology, our proposed attacker reconfigures sensor chips via shared buses, as they are generally not protected by security mechanisms. We investigate how such sensor deprivation impacts the control of systems such as drones, how applied embedded control platforms practically react to SDAs, and find that those attacks can have unexpected consequences leading to a complete stop of control computations. As a result, the process will not be stabilized any longer, and, e.g., the drone will crash or deviate. We show that SDAs that the proposed attacks are undetected from recently proposed UAV anomaly detectors. Finally, we propose an attack synthesis methodology to optimize the timing of such SDA manipulations to deterministically deviate the drone from the planned trajectory.

- In Chapter 5, we present the goal and approach for the security analysis of anomaly detection for CPS. We start by providing a taxonomy of process-based detectors. Then we formally define the system and attacker model considered to evaluate the target anomaly detection systems. Specifically, we propose concealment attacks and constraints on the attacker.

- In Chapter 6, we investigate different approaches to evade prior-work reconstruction-based anomaly detectors (i.e., deep learning models) by manipulating sensor data so that the physical anomalies are concealed. We find that replay attacks (commonly assumed to be very strong) show bad performance (i.e., increasing the number of alarms) if the attacker is constrained to manipulate less than 95% of all features in the system, as hidden correlations between the features are not replicated well. To address this, we propose two novel attacks that manipulate a subset of the sensor readings, leveraging learned physical constraints of the system. Our attacks feature two different attacker models: A white-box attacker, performing an iterative optimization attack, and a black-box attacker, which uses an adversarially trained generator to generate the adversarial examples.

- In Chapter 7, we hypothesize that the detectors verify a combination of temporal, spatial, and statistical consistencies. To test this, we systematically analyze their resilience against generic concealment attacks. Our generic concealment attacks are designed to violate a specific consistency verified by the detector and require no knowledge of the attacked physical process or the detector. In addition, we compare against the attacks from Chapter 6 that were designed to attack neural network-based detectors. Our results demonstrate that the evaluated model-free detectors are in general susceptible to generic concealment attacks. For each evaluated detector, at least one of our generic concealment attacks performs better than Chapter 6 attacks. In particular, the results allow us to show which specific consistencies are verified by each detector. We also find that the attacks presented in Chapter 6 that target neural-network architectures transfer surprisingly well against other architectures.

- In Chapter 8, we demonstrate for the first time that white-box concealment attacks can also be applied to detectors that are not based on neural network solutions. To achieve this, we propose a generic white-box attack that evades anomaly detectors and can be adapted even if the target detection technique does not optimize a loss function. We design and implement a framework to perform our attacks and test it on several detectors from related work. Our results show that it is possible to completely evade a wide range of detectors (based on diverse detection techniques) while reducing the number of samples that need to be manipulated (compared to black-box concealment attacks proposed in Chapter 7).

1.2 Thesis Reproducibility

Each of the research papers collected in this thesis is complemented with the source code and the data to reproduce the results. Notably, when organized by the conference, we also submitted the code for revision by an artifact evaluation committee. The artifacts of two of the research papers [P1, P3] were reviewed by such a committee. For the paper [P3], we received the *Distinguished Paper with Artifact Award*. All the artifacts related to the content presented in this thesis are available online:

- Chapter 3—`https://github.com/scy-phy/OPC-UA-attacks-POC`

- Chapter 4 https://github.com/scy-phy/droneDeprivation
- Chapter 6—https://github.com/scy-phy/ICS-Evasion-Attacks
- Chapter 7—https://github.com/scy-phy/ICS_Generic_Concealment_Attacks
- Chapter 8—https://github.com/scy-phy/whiteboxDimva23

2
Background

Cyber-Physical Systems and Adversarial Machine Learning

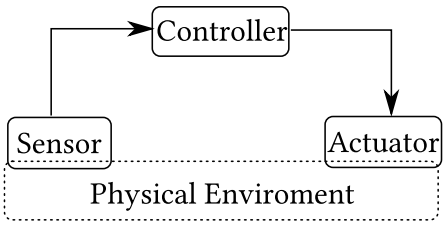


Figure 2.1: Cyber-Physical System architecture, the control loop.

2.1 Cyber-Physical Systems

The term Cyber-Physical Systems (CPS) refers to systems where physical and computational components interact to accomplish a task in the physical world. Cyber-Physical Systems encompass a wide spectrum of applications [119]. Examples of such systems include Autonomous Vehicles, Robotic and Unmanned Aerial Vehicles, and Industrial Control systems. In this chapter, we present the relevant background information for this thesis.

2.1.1 Cyber-Physical Systems Architecture

The general CPS architecture consists of three main components which together form the Control Loop (see Figure 2.1). Sensors: measure the physical environment; controllers: use the information received from sensors and decide which actions to take; and actuators: execute those commands. The various cyber-physical systems, which share the same general architecture, differ from one another at the implementation level (i.e., hardware, software, and network). Each is characterized by design choices imposed by the different nature of the tasks they accomplish. For this thesis, we describe in detail the common architectures adopted for Industrial Control Systems and Unmanned Aerial Vehicles.

2.1.1.1 Industrial Control Systems

Industrial Control Systems (ICS) are widely used to automate processes in production plants and facilities. ICSs are composed of interconnected Cyber and Physical components to interact with the physical environment. Cyber components comprise the hardware and software that are used to control the process. The Purdue Enterprise Reference Architecture (PERA) [205] is the networking architecture for ICS systems, adopted in the ANSI/ISA-95 standard (Figure 2.2). The PERA model divides the ICS system into six layers, the layers from 0 to 3 constitute the manufacturing zone, while levels 4 and 5 constitute the enterprise zone.

ICS processes are constituted by sub-processes (e.g., electrical substations). To control the different sub-processes, we can divide the PERA architecture into areas. Each area is composed of levels 0 to 2. At level 0, the sensors and actuators are deployed to interact with the physical process. At level 1, Programmable Logic Controllers (PLC) are deployed to implement the system's control logic (e.g., Proportional Integral Derivative (PID) controller). PLCs observe sensor values and send commands to actuators. At level 2, local Supervisory Control and Data Acquisition (SCADA) systems are deployed

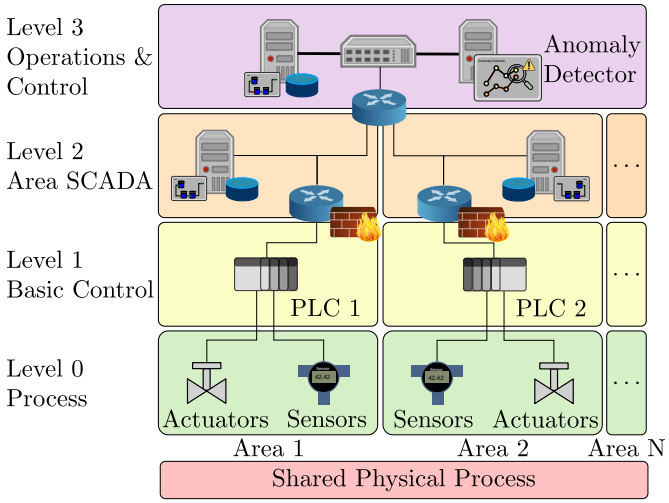


Figure 2.2: Purdue Enterprise Reference Architecture (PERA) (205), for Industrial Control Systems

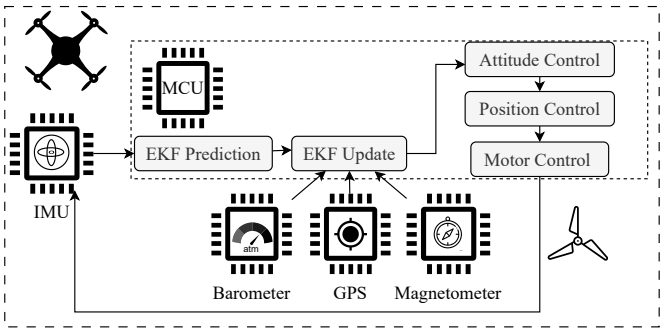


Figure 2.3: Flight control architecture—sensors, GPS receivers and their contribution to the state estimation.

for Area supervisory purposes. At level 3, the different areas are aggregated to perform plant monitoring. The aggregation can happen locally for co-located areas (i.e., through a router), or via the Internet/WAN for distributed sites. Monitoring can occur through Human Machine Interfaces (HMIs) and anomaly detectors.

A variety of network protocols are commonly found in such architectures. Between level 0 and level 1, serial communication is used to communicate between PLCs and sensors and actuators. Between level 1, level 2, and level 3 and within levels 1, 2, and 3 industrial protocols such as EtherNet/IP, PROFINET, or OPC UA are used to communicate. The choice of the protocol depends on the protocols supported by the industrial devices in the network.

2.1.1.2 Unmanned Aerial Vehicles

Autonomous UAVs are composed of sensors and actuators connected to a flight controller (Figure 2.3). Sensors observe the UAV state and report it to the flight controller. The

flight controller is equipped with a microcontroller unit which is used to run the control and communication software. The received sensor readings are then used for state estimation via sensor fusion algorithms (e.g., Kalman Filter). Based on the estimated state and the planned trajectory, the flight controller computes the control action, which is finally transmitted to the actuators (motors with propellers).

Sensors for autonomous navigation. Inertial Measurement Unit (IMU) is an essential sensor for flight stabilization. It uses an accelerometer and gyroscope to report the proper acceleration and the angular velocity to the flight controller, which utilizes this information for attitude estimation (estimating the pose of the drone). Other sensors required for autonomous navigation are compasses, magnetometers, and barometers. Moreover, a GPS receiver is necessary to locate the drone. Readings from all these sensors and the GPS receiver are used as input to the Kalman Filter for state estimation.

EKF architecture. Extended Kalman Filter is the non-linear extension of the Kalman Filter [104]. The Extended Kalman filter is used for the estimate of unknown variables by fusing multiple sensor information (this technique is also referred to as sensor fusion or software sensing). It operates in two steps, the prediction step and the update. During the prediction step, the state estimate is computed (e.g., roll, pitch, yaw, speed of the UAV), while during the update the predicted state is improved by fusion with other sources.

Popular open-source flight control software such as Ardupilot applies state-of-the-art Extended Kalman Filter state estimation for autonomous navigation. According to the documentation [13], state prediction relies on IMU data, while the state update is done by fusing the other available sensor readings (e.g., GPS position, barometer, and magnetometer). State prediction is calculated every time new IMU data is received (IMU has a high sampling rate), while state update is performed less frequently (as the other sensors are sampled less often). The IMU sensor reading is the most important and with the highest priority in the state estimation procedure.

2.1.2 Cyber-Physical Systems Security

CPS are pervasive in our society, and for this reason, their security is relevant. CPS security presents unique challenges in its deployment. Given the high degree of interconnections in a CPS, the overall security of CPS deployments relies on trustworthy communication. In practice, CPS systems are often deployed relying on protocols that do not implement security features (such as authentication or encryption) e.g., Fieldbus [70], CAN [70], or mavlink [109]. Legacy compliance is one of the reasons for the absence of security in CPS, CPS often relies on obsolete but expensive components, which were designed for air-gapped systems, that do not support secure communication. Another reason for the absence of security is that CPS often relies on resource-constrained devices (e.g., microcontrollers), and embedding secure communication will induce an increased workload that undermines the real-time requirements of the systems. Communication protocols that promise security were introduced for ICS, but in practice, there are challenges in deploying secure CPS [49].

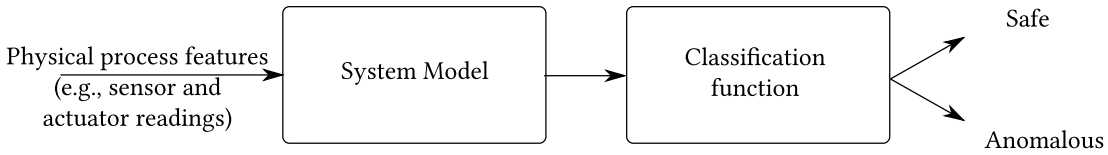


Figure 2.4: Process-based anomaly detection, general architecture.

2.1.2.1 Attacks to CPS

Given the importance of CPS for our society, they are a valuable target of attacks [31]. Real-world incidents targeting CPS occurred in the past. For example, attacks to ICS and critical infrastructures e.g., Stuxnet [202] targeting nuclear plants, the Colonial Pipeline attack [204] targeting gasoline pipeline and Oldsmar's water treatment attack [37] targeting a water facility. Moreover, attacks on vehicles have occurred, for example, Miller and Valasek [132] demonstrated remote attacks against unaltered cars gaining access to the car network and remotely conducting the car out of the road.

The common goal of attacks is to physically or remotely exploit the CPS to cause process disruption or interfere with its tasks (e.g., induce defect in produced goods in an ICS, cause sub-optimal control decisions that induce wrong actuation, and destabilize the process). Attacks against CPS can occur at various levels of its architecture according to attacker capabilities, via network attacks, physical attacks to the actuator and sensors, and attacks to the control algorithm. Attacks can occur locally by deploying the attack within the CPS, or remotely, by exploiting wireless communication.

Network attacks. The attacker tampers with the sensor and actuator messages in the network to cause process manipulation. Examples of such attacks are Denial of Service (DoS) attacks where the attacker drops all the sensor or actuator messages, False data injection attacks where the attacker manipulates the network messages to cause wrong system estimation (when manipulating the sensor readings) or wrong actuation (when manipulating the actuator commands). Network attacks represent a threat to CPS networks that rely on legacy protocols that do not support authentication or encryption [70].

Physical attacks. The attacker can physically tamper sensors and actuators to prevent correct sensing or actuation. Examples of such attacks range from physically breaking sensors/actuators to interfering with their sensing and actuation capabilities by acting in the physical environment. The physical interference with sensors exploits their inner functioning to induce unwanted effects. Examples of physical interference with sensors are sound-induced vibrations against Inertial Measurement Units (IMU) [180], laser-induced sound commands against microphones [183], LIDAR sensor spoofing [30], light-based attack against depth estimation [213].

Control Algorithm attacks. The attacker modifies the control program running in the CPS controller. This can be obtained by inserting malicious in the target controller which is crafted to reach the attacker's desired behavior on the victim system via logic bombs [81] or Man-in-the-PLC [71].

2.1.2.2 Anomaly Detection in CPS

Intrusion Detection refers to software techniques used in computer systems to identify malicious activities in the network [66]. Traditional Intrusion Detection for IT networks relies on the analysis of network features for the detection. In the context of Cyber-Physical Systems, Intrusion Detection was proposed to detect attacks that target critical assets [32, 195]. Given the nature of CPS, not only network features are used to detect intrusions but also physical process features can be leveraged to detect the anomalous patterns in sensor readings caused by an attack (referred to as Process-based Anomaly Detection).

Anomaly detection is a binary classification problem, where the goal is to classify if the sensor readings coming from the CPS process are 'safe' or 'anomalous'. An anomaly detector (Figure 2.4) is usually composed of a System model that models the behavior of the CPS and a classification function (for example the system model is trained with Machine-learning techniques). The general idea is to input the sensor readings into the system model, which is used to predict how sensor readings will evolve. Then the classification function decides if the system is under attack (by monitoring the difference between the observed sensor readings and the actual sensor readings (residual)). In the literature many techniques have been proposed to perform anomaly detection, in Part II we provide details and a taxonomy of anomaly detection for ICS.

2.2 Adversarial Machine Learning

Adversarial machine learning is the research field that studies the security of machine learning applications [23]. The development of a machine learning application, involves various stages, data collection, classifier training, classifier evaluation, and classifier deployment. Machine learning pipelines are vulnerable to attacks, and those attacks aim at perturbing the machine learning classifier's performance. Attacks on the machine learning classifier can occur in any stage of the pipeline where untrusted input is used. A detailed discussion of attacks on machine learning classifiers and their taxonomy is available in the literature [155, 97, 18, 78]. For this thesis, we summarize the attacks that can occur on machine learning pipelines.

Biggio et al. [23] provide a taxonomy of machine learning models. Attacks can be differentiated between training-time attacks, test-time attacks, and train-test attacks. Attacks that occur over training time are called *poisoning* attacks, attacks that occur over testing time are referred to as *evasion attacks*, and attacks that occur both during train and testing are called *backdoor* attacks. Moreover, attacks against classifiers can be categorized as targeted or untargeted. In targeted attacks, the adversary aims to control the output of a machine learning classifier and obtain a specific classification outcome. In untargeted attacks, the attackers aim to get a sample misclassified without specifying the target class.

Poisoning attacks. Data poisoning exploits the use of untrusted or unverified data used for classifier training. In a data poisoning attack, the attacker modifies the training data or their associated ground truth, to achieve misclassification. The resulting trained classifier learned the wrong patterns associated with the attacked class, and the

functionality of the classifier on the attacked class is not preserved.

Evasion attacks. In an evasion attack, the attacker does not influence the training process of the classifier, hence the classifier functionality is not influenced. Instead, the attacker manipulates the input data that are used by a deployed classifier. The adversarial controlled input will produce the misclassification. The perturbed sample is called *adversarial example*.

Backdoor attacks. Backdoor attacks operate in two steps. The first step occurs during classifier training. The attacker modifies the training data to embed a new functionality on the victim classifier (the backdoor). The second step occurs at testing time, to activate the implanted functionality the attacker uses a trigger. When the trigger is present in the input, the classifier for example will produce misclassification. In a backdoor attack, the classifier functionality is preserved unless the trigger is present in the input.

In Part II of this Thesis, we formally define evasion attacks in the context of CPS and evaluate the performance of process-based anomaly detectors for cyber-physical systems against classifier evasion.

Part I

Security Issues in State of the Art Cyber-Physical System

3

Modern Industrial Protocol Security

Security Analysis of Vendor Implementations of the OPC UA Protocol for Industrial Control Systems

3.1 Introduction

The increasing interconnection of industrial components critically relies on the security of the communication protocol adopted for Machine to Machine (M2M) communication to prevent adversarial manipulation of the process, leading to significant monetary loss and physical damage [164]. The OPC Unified Architecture (OPC UA) protocol [64] is often considered the de-facto standard for building Industry 4.0 processes and it is the reference communication protocol in RAMI4.0 (the reference model for Industry 4.0 [215] in European countries such as France, Italy and Germany [69]). In 2006, the OPC Foundation released the first OPC UA specification featuring security mechanisms such as authentication, authorization, integrity, and confidentiality. The German Federal Office for Information Security (BSI) released the 'OPC UA Security Analysis' [27], reporting that 'No systematic errors could be detected' in the OPC UA standard.

Nevertheless, there seem to be significant challenges to set up secure OPC UA deployments in practice. In [49], the authors perform a scan for OPC UA servers reachable over the Internet and find that 92% of the deployments show issues with security configuration. Among those, 44% of the servers are accessible without any authentication requirements. To mitigate that issue, the authors suggest (similar to [27]) to disable insecure security modes and policies and enforce user authentication. The authors of [49] suggest that their findings are due to the configuration complexity of OPC UA, but no root cause analysis was provided. Indeed, OPC UA requires a correct configuration to prevent attacks such as: i) attackers feeding incorrect information to clients (Rogue Server attack); ii) eavesdrop and change values which can directly alter the physical process (Rogue Client attack); iii) or both (Middleperson attack).

Security threats through configuration complexity are in general investigated in the field of usable security [4, 83, 115]. For example, studies demonstrated that a little set of demo applications or poor documentation resulted in insecure deployments [115]. Challenges in setting up OPC UA networks were not investigated so far.

In this work, we are the first to systematically investigate the security of a range of OPC UA libraries and products. Despite the popularity of OPC UA and its advanced security concept (compared to other industrial protocols), we show that in practice many OPC UA artifacts have missing support for security features of the protocol. Those limitations make the security configuration infeasible, or give a false sense of security (i.e., traffic encryption occurs with untrusted parties). Several of the artifacts that claim to feature OPC UA security rely on libraries that are incomplete, insecure and provide instructions that lead to insecure settings. While such issues are implementation-specific, we show that the OPC UA standard is insufficient, as it supports those insecure behaviors. Lastly, we propose several practical countermeasures.

To assess the vulnerability of libraries, we create a framework that implements Rogue Server, Rogue Client, and Middleperson attacks. For each library, we set up a client and/or server application using protocol options that can be expected to provide a secured OPC UA instance, and then attack that system. For cases where the attacks succeed, we further investigate the cause. We show that Middleperson attacks

are possible and allow to manipulate the process and prevent reliable control of the system. Surprisingly, we find that even recovery of plaintext credentials from intercepted encrypted communications is easily possible with our Rogue Server and Middleperson attacks. This finding contrasts with prior work [157], where the authors perform a Middleperson attack on OPC UA applications and conclude that it is impossible to recover user passwords in OPC UA systems (even with security mode 'None').
Contributions. We summarize our contributions as follows

- We systematically analyze 22 products and identify significant recurring (for 15 out of 22) issues with the availability OPC UA security features, or their setup instructions.
- We systematically analyze 16 libraries and for all of them we identify at least one reoccurring issue with the implementation of security features (either at client side, at server side or both).
- We demonstrate the feasibility of the identified attacks by implementing the attacks with custom proof-of-concept applications[1].

Disclosure. We discussed our findings with the OPC foundation, leading to improvements of the specifications. Our work was based on publicly available information extracted from user manuals, library documentation and the OPC UA standard. As the feasibility of our attacks depends on (past) end-user configuration process carried out in the target system and not on a specific software/hardware vulnerability, there was no disclosure to vendors required. Nevertheless, our findings already led to improvements of at least one artifact after public discussion of this work.
Organization. The remainder of this work is organized as follows. In Section 3.2 we present the details of the OPC UA protocol. In Section 3.3 we present our research methodology together with system and attacker model. In Section 3.4 we explain how we build the framework that we use to analyze the libraries. In Section 3.5 we report the results of our research. A discussion of countermeasures is presented in Section 3.6. Related work is presented in Section 3.7 followed by conclusions in Section 3.8.

3.2 Background on OPC UA

OPC UA [64] is an industrial communication protocol developed by the OPC foundation that allows platform-independent and secure communication by design. Those two characteristics make this protocol stand out when compared to other common industrial protocols. Adopting this protocol in an industrial plant allows integration between heterogeneous hardware, which is instead a constraint brought by proprietary protocols (e.g., Siemens S7). At the same time, the modern *security-by-design* features offered by the protocol promise to mitigate some of the common security threats for CPS (e.g., no message authentication or encryption). The last OPC UA specification 1.04 was released in 2018 and is divided into eighteen parts. Part 2 describes the security

[1]The framework is available at https://github.com/scy-phy/OPC-UA-attacks-POC

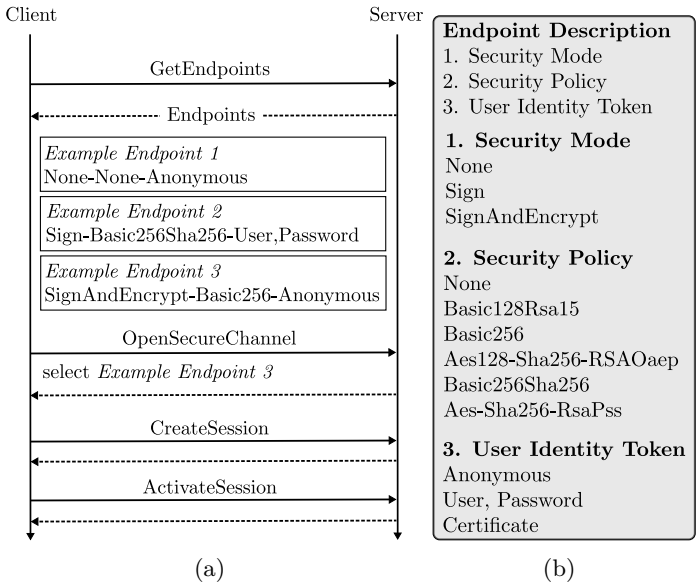


Figure 3.1: (a) Overview of the connection establishment procedure. The server provides a list of endpoints. The client selects which endpoint to connect to. If a secure endpoint is chosen, the application authentication through certificates is performed. (b) Options available for the endpoint configuration. The Security Mode, Security Policy and User Identity Token are chosen independently from each other.

model of OPC UA, and Part 4 describes the services that implement security primitives. Two communication strategies are possible: client-server and publisher-subscriber. Both allow messages to be signed to ensure authenticity and encrypted to add confidentiality. OPC UA does not enforce the use of security, messages can be exchanged without security. Figure 3.1(a)(b) reports an example of secure connection establishment in client-server. A server offers several endpoints, each defined by a Security Mode, a Security Policy, and the supported User Identity Token(s). When initiating a connection, the client chooses the endpoint (from an unauthenticated list of endpoints). The Security Mode defines how messages are exchanged to achieve authentication, confidentiality, and integrity. Available Security Modes are None, Sign, SignAndEncrypt. Security Policies define the cryptographic primitives used for the specific security mode. The UserIdentityToken defines the supported user authentication methods for an endpoint: Anonymous (no user authentication), Username&Password, Certificate [146].

Certificate Management. Secure connection establishment (Security Mode not 'None') requires OPC UA applications (clients and servers) to exchange their Application Instance Certificate. Upon receiving a certificate, an application needs to decide whether the received certificate is trustworthy. To this end, each OPC UA application has a list of certificates that are trusted (Certificate Trustlist). This list contains self-signed certificates or certificates from Certificate Authorities (CA). In the first case, the certificates are often exchanged manually between applications, but the use of a Certificate Manager with a Global Discovery Server (GDS) is also possible. GDS provides two main functionalities: i) Application discovery: OPC UA applications use

the GDS to find available applications that have previously registered with the GDS. ii) Certificate management: applications push and pull certificate to the GDS to update Trustlist.

Secure Connection Establishment. To establish a secure connection, the server sends its certificate through the `GetEndpoints` response. If the client trusts the server and selects a secure endpoint, the `OpenSecureChannel` message carries the client certificate. Upon trusting the client's certificate on server side, the secure channel is established. The `OpenSecureChannel` request and response use asymmetric encryption and messages contain nonces used to compute the symmetric signing and encryption keys used in sessions [145].

Service Sets. OPC UA offers its functionalities through 10 service sets. Each service consists of a Service request and Service response, identifiable via RequestHeader and ResponseHeader. The ten service sets (and therefore the functionalities) offered by the protocol are Discovery, SecureChannel, Session, NodeManagement, View, Query, Attribute, Method, MonitoredItem, and Subscription. Each service set specifies a different functionality as defined in Part 4 of the OPC UA protocol [144]. For example, a client can read and write values related to the processes on OPC UA Nodes in the server through the Attribute service set. It is also possible to provide the functionality for the clients to add nodes. Moreover, clients can call methods (Method service set) defined for Object Nodes. Through these functionalities, a client can directly influence the plant operations. The OPC UA server authorizes clients to use its services via user authentication. Session service set provides the user authentication functionalities. User authentication occurs during session activation, after the initial application authentication.

3.3 OPC UA Security Assessment Methodology

OPC UA features a number of cryptographic options to establish secure channels (see Section 3.2). Like in any system, authentication of parties to each other (application authentication in OPC UA) requires either pre-shared secrets or certificates with public keys. In this work, we argue that in the ICS setting (in which OPC UA is adopted), no public PKI (e.g., with root CAs) is available, so a core issue is how to establish the initial trust between parties. On the Internet, the initial distribution of public keys is commonly solved by shipping devices (or OS) with certificates of a set of core root certificate authorities (CAs). Servers (identified by unique DNS names) then provide certificates authenticated directly (or indirectly) by those root CAs. Such a solution is not possible for ICS networks, as they are air-gapped and do not provide (externally verifiable) unique addressing. Local self-signed CAs are an alternative, but their certificates will not be shipped together with devices newly introduced into the system. For this reason, bootstrapping the security in an OPC UA system relies on the manual certificates distribution. In OPC UA there are two ways to perform certificate management and distribution: i) Through self-signed certificates, ii) through the GDS *CertificateManager* (see Section 3.2). The focus of our security assessment is the application authentication functionality required for the security in OPC UA.

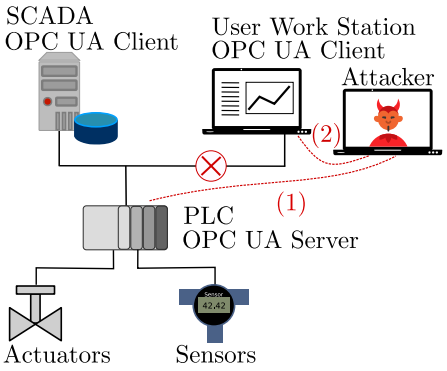


Figure 3.2: Example OPC UA network with a server and two clients. An attacker that wants to establish themselves as a PitM has to (1) pose as an OPC UA client towards the legitimate OPC UA server and (2) as an OPC UA server towards a legitimate client.

3.3.1 System and Attacker Model

We assume a local ICS network with OPC UA server as depicted in Figure 3.2. The system operator guarantees security in the OPC UA system by allowing the server to offer only secure endpoints (i.e., Sign or SignAndEncrypt, see Section 3.2) as suggested by the BSI [27]. We note that Intrusion Detection Systems are out of scope in our model as we aim to look at security guarantees from the OPC UA artifacts. We assume the network operator follows the user manual shipped with the deployed ICS hardware/software to configure the OPC UA server and client security (this is a common assumption in related usable security works, e.g., [4]). In this work, we consider three attackers with different goals corresponding to the OPC UA model introduced in the Part 2 of the standard [146].

Rogue Server. A new device is introduced and now needs to establish secure OPC UA communications with other devices. An attacker is present in the network and aims to manipulate OPC UA clients by providing malicious information or stealing OPC UA user credentials. They[2] create a server that offers secure endpoints to establish a secure connection with new clients and make them believe that they are communicating with the actual OPC UA server in the network.

Rogue Client. An attacker aims to connect to the OPC UA server to eavesdrop or manipulate the information shared between the server and clients. They create a client that attempts to connect to the server although not authorized by the network operator.

Middleperson attack (PitM). An attacker aims to establish themselves as Middleperson between the client and server, intercepting and manipulating all communications between both. This requires achieving Rogue Client and Server objectives.

[2]We use the gender-neutral 'they' as pronoun in this work

3.3.2 Research Questions and Challenges

With our research, we aim to address the following research questions. **RQ3.1** *What are practical challenges for the correct use of OPC UA security features?* **RQ3.2** *Are OPC UA security features correctly implemented by the vendors and products?* **RQ3.3** *What are the consequences of breaking OPC UA security features?*

While addressing those research questions, we tackle the following research challenges: i) Partially proprietary products without source code. ii) available OPC UA libraries partially documented, iii) unavailability of products (or real deployments) for testing.

3.3.3 Proposed Approach

Our approach focuses on the analysis of security features implemented by OPC UA compatible artifacts. We focus on implemented key management functionalities i.e., i) Proprietary hardware and software that implements OPC UA stack, ii) Open-source libraries that implement the OPC UA stack.

To address RQ3.1, we survey proprietary and open source OPC UA enabled products. We investigate practical challenges for correctly configuring secure OPC UA setups in the analyzed products checking availability of features. Investigating challenges to set up security is common practice in usable security, where resources available to developers are analyzed to understand if they help configuring secure systems [4, 83, 115, 114].

To address RQ3.2, we propose a framework to verify the correctness of implementations of OPC UA security features in hardware and software products. For proprietary products, we consult user manuals as we had no access to physical products. For libraries, we practically tested the security features deploying OPC UA clients and server locally. We tested the three attacks presented in the Attacker Model, which are feasible due to erroneous or incomplete key management features. For each library, we set up OPC UA server and client following the instructions and demo applications provided with libraries. Testing the application is useful because, a) not all functionality provided by products was sufficiently described in documentation to be able to classify its features for our work; b) documentation and code does not always match, so we double-checked with our implementation and demos. We note that, demo applications, user manuals and documentations are resources used by developers to build their applications, so it is realistic to follow them to identify poor security implementations and propagation of security pitfalls [4, 115]. Recently, it was reported that it was possible to alter the infotainment system firmware on a 2021 Hyundai Ioniq car using publicly known AES 128-bit CBC private keys provided as an example in NIST documents [190].

To address RQ3.3, we show that breaking the application authentication configuration will have severe consequences on OPC UA security. An attacker will be able to obtain plain-text passwords when legitimate clients connect to Rogue Server or the Middleperson system and perform user authentication, to modify the data seen by OPC UA clients and servers, and to execute function calls on an OPC UA server that can interfere with the physical process (e.g., send commands to actuators).

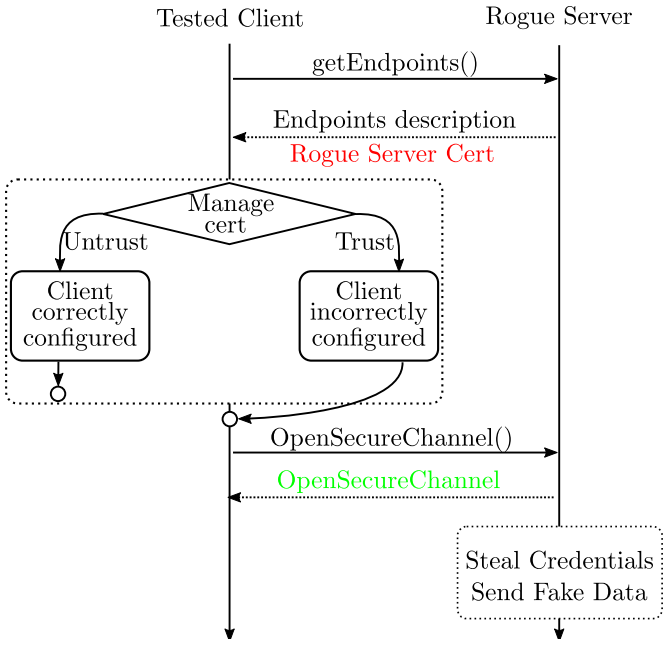


Figure 3.3: Rogue Server workflow. The server waits for a new connection. When the Server receives a getEndpoints request it provides a certificate that was not trusted on the client side upon connection (in red). If the client trusts the server certificate and sends an OpenSecureChannel to the server, the Rogue Server allows the connection (in green). The attacker can, for example, steal credentials and send manipulated data.

3.4 Framework and Implementation

In this section, we present our framework for the security assessment of OPC UA artifacts. First, we will describe our framework from a high-level perspective. Then, we provide the details of our framework implementation.

3.4.1 Framework

Our framework tests OPC UA clients and OPC UA servers against Rogue Server, Rogue Client, and Middleperson attacks. For OPC UA servers, the framework identifies vulnerabilities to Rogue Client Attacks, while for OPC UA clients the framework identifies vulnerabilities to Rogue Server Attacks. The framework checks for vulnerabilities generated by incorrect (or incomplete) Trustlist management. When both Rogue Client and Rogue Server vulnerabilities are present in the OPC UA network, Middleperson attacks can be exploited.

Rogue Server. In Figure 3.3 we report the steps followed to verify the vulnerability to the Rogue Server. The test employs an OPC UA Rogue Server to test OPC UA clients. The Rogue Server waits for a `getEndpoints` request from a victim client and replies offering secure endpoints and its self-signed application instance certificate that is not present in the client's Trustlist. The victim client receives the endpoints and the certificate. The Trustlist of the OPC UA victim client does not contain the Rogue

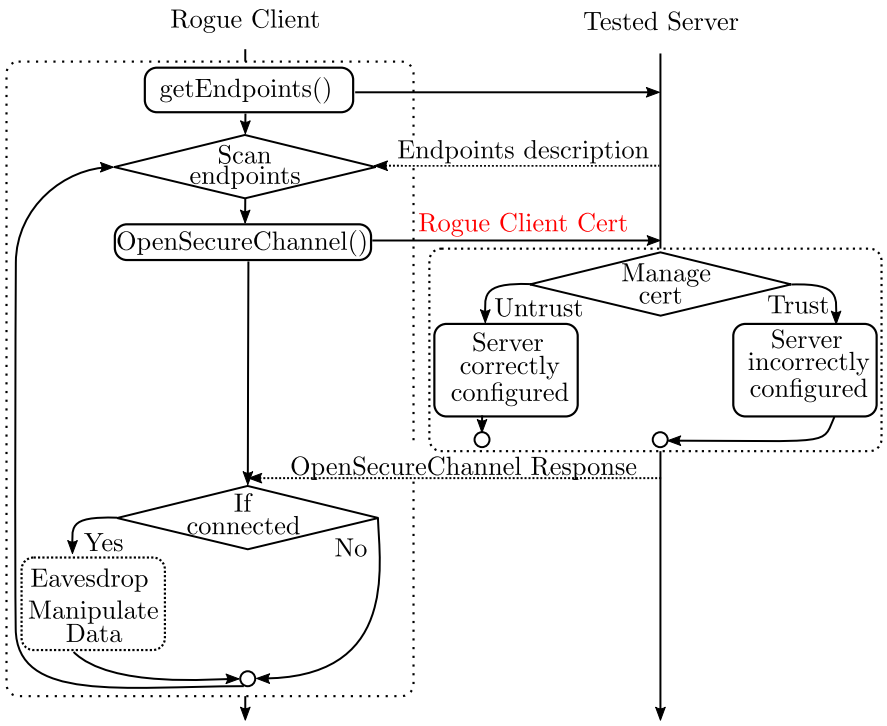


Figure 3.4: Rogue Client workflow. The dashed boxes contain the flowchart representing the application server/client logic. Diamonds represent a decision by the client/server according to the data flow and management. The client lists all the secure endpoints on the server and for each one, it tries to connect providing an arbitrary self-signed certificate (highlighted in red) that was not shared with the server in advance. If the client successfully connects, the server is not correctly managing certificates.

Server's certificate. The victim client should not establish a secure connection with the untrusted Rogue Server as the root of trust between client and Rogue Server was not established upon connection. If the victim client is correctly configured, it will not continue the interaction with the Rogue Server. Otherwise, the victim client trusts the Rogue Server and instantiates an `OpenSecureChannel` request that the Rogue Server accepts.

As a consequence, an attacker can send fake data to the victim client and steal user credentials. User credentials stealing occurs when the `ActivateSession` request with 'UserIdentityToken' 'user&password' is instantiated by the Client. The request (containing user and password) is encrypted with the keys derived from the `OpenSecureChannel` Nonces. Moreover, the password is encoded in UTF-8 and encrypted with the server public key (i.e., the key received from the untrusted Rogue Server). Despite the different encryption techniques applied to encrypt the password before transmission, the Rogue Server will decrypt them because the client allows the connection with untrusted parties.

Rogue Client. In Figure 3.4 we report the steps followed by our framework to verify the vulnerability to the Rogue Client in OPC UA servers. The test employs an OPC UA Rogue Client to the test OPC UA server implementations. The Rogue Client scans all

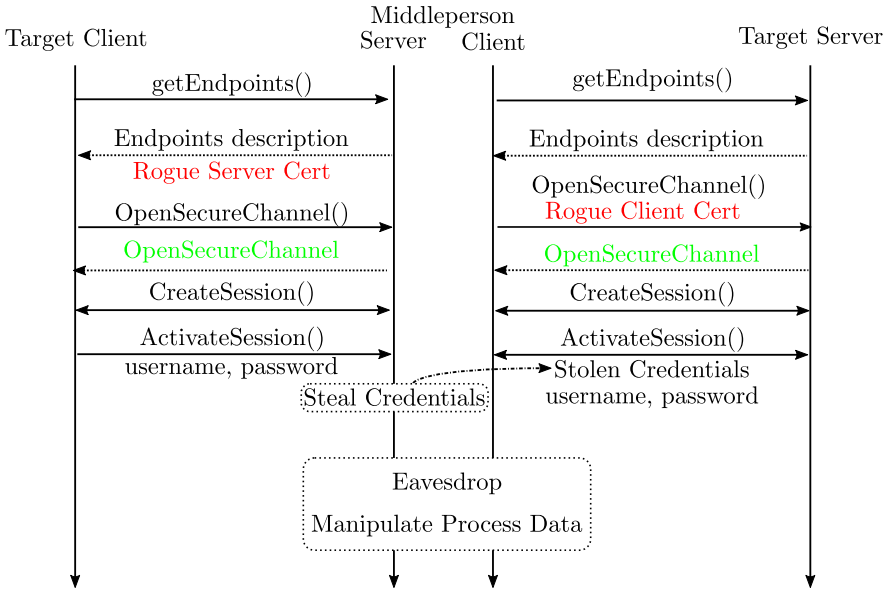


Figure 3.5: Example of Middleperson attack with the stealing of session credentials (User Identity Token: User&Password). The attacker acts both as Rogue Client and Server. The Target Client connects to the Middleperson Server. The client is vulnerable to Rogue Server attack and trusts the server cert without verification and instantiates a OpenSecureChannel() request. The Rogue Server allows the connection. When the client instantiates a session the Rogue Server accepts it and decrypts the provided credentials. The Middleperson connects to the Target Server (vulnerable to Rogue Client Attacks) and creates a session providing the stolen credentials

the endpoints offered by an OPC UA server and tries to establish a secure connection to all of them, one by one. The Trustlist of the OPC UA server does not contain the Rogue Client's certificate. Hence, the Rogue Client should not be entitled to establish a secure connection with the server because the root of trust between client and server was not acknowledged upon connection. If the client succeeds in connecting, it means that the server implementation is managing erroneously the certificates, and it is deviating from the expected behavior prescribed by OPC UA protocol. The server is then considered vulnerable to Rogue Client attacks.

As a consequence of a Rogue Client attack, an attacker can perform different actions on the server according to the server configuration. The attacker possibilities range from reading values published by the attacked OPC UA server to writing values and executing commands on the server that influence the physical process.

Middleperson. Figure 3.5 reports the steps followed by our framework to verify the vulnerability to the Middleperson attack. The test leverages the concepts of Rogue Client and Rogue Server used at the same time to achieve the Middleperson attack. The attacker instantiates a Rogue Server in the network. The client requests a secure connection with the Rogue Server (as described in the previous paragraphs). If the attacked server requires user authentication with username and password, the Rogue Server can request the user identity token from the victim's client and steal the credentials. At this point, the Rogue Client instantiates a connection with the victim's server by providing the

stolen credentials and takes control of the industrial process (as described in the Rogue Client paragraph). The attacker forwards stealthily (from the process operators) the information received from the victim client to the victim server and vice versa.

Consequences of the Middleperson attack comprise the union of the outcomes identified for Rogue Client and Rogue Server.

3.4.2 Implementation

We implemented our framework (i.e., Rogue Server, Rogue Client, and Middleperson) using the python opc-ua [67] open-source library. The implementation serves as a Proof of concept (POC) of our attacks. We have implemented the POC using the python opc-ua since the library offers support to all the functions required to implement our proposed attacks. Moreover, the functionalities are easy to prototype and deploy.

Our framework consists of four python modules, `framework`, `utils`, `rogueclient`, and `rogueserver` for a total of 571 lines of code. For each of the three attacks, we report a description of how we realized functionalities through the python modules.

Framework module. The `framework` module interacts with the attacker. It offers a command-line interface to select the attack (i.e., Rogue Server, Rogue Client, or Middleperson).

Rogue Server attack. When Rogue Server attack is selected, the `rogueserver` module performs the following to mount and start the attack. First, the program executes port scanning in the network to identify available OPC UA servers on port 4840. The attacker selects a benign server that they want to clone with the Rogue Server attack. At this point, the sequence of unencrypted and unauthenticated primitives `FindServers()` and `GetEndpoints()` requests are sent to the benign server to retrieve endpoints information, server information, and the server certificate. A Rogue Server is created with this information. The Rogue Server has the same name, offers the same endpoints (same security mode, security policy), the same user identity token, and provides a self-signed certificate filled with the same information as the benign server (except fingerprints). When Rogue Server is configured, port forwarding is enabled in the network to route requests intended to the victim's server to the Rogue Server. Finally, the server is started and waits for a victim client to connect to it without performing certificate validation. If the connection succeeds, the Rogue Server starts publishing fake data, and if the victim's client provides user credentials, the Rogue Client decrypts them.

Rogue Client attack. When Rogue Client attack is selected, the `rogueclient` module performs the following actions to mount and start the attack. First, the attack performs port scanning in the network to identify available OPC UA servers on port 4840. The attacker chooses the victim server that they want to connect through the Rogue Client attack. At this point, the client attempts the `OpenSecureChannel()` request to the victim server endpoints (one by one), providing a self-signed certificate that was not inserted in the victim server Trustlist upon connection. If the cert is trusted, the attack is successful. Furthermore, if the victim server requires the user identity token 'None': the Rogue Client instantiates a `ActivateSession()` request and can start reading (or writing) values at server nodes. If a user identity token with

username and password is required, the attacker can input them (e.g., they retrieved them with a Rogue Server attack). Finally, the `ActivateSession()` request is sent to the server.

Middleperson attack. When the Middleperson attack is selected, our POC realizes the attack depicted in Figure 3.5. The `rogueserver` and `rogueclient` modules are used in parallel (i.e., in threads). First, the Rogue Server is created to capture victim client requests in the network and acquire user credentials. Once the credentials are retrieved, the Rogue Client is instantiated and connects to the victim's server providing the stolen credentials. The attack is successful if both the victim's client and server do not populate the Trustlist upon connection.

3.5 Assessment Results

In this section, we present the results of our security assessment of 22 OPC UA products by different vendors (i.e., mostly OPC UA servers for PLCs), and 26 systems we build using 16 libraries. We start by reporting the selection criteria of the artifacts considered in this work. Next, we investigate the availability of features of the OPC UA stack implemented in the considered artifacts. Then we perform a security assessment for OPC UA products with our framework and look for vulnerabilities to Rogue Client, Rogue Server, Middleperson attacks.

3.5.1 Artifacts Selection Criteria

For the two categories of OPC UA artifacts considered in our work, we used the following selection criteria. Note that not all the artifacts are certified by the OPC Foundation.

Proprietary products. We assessed only proprietary products with publicly accessible manuals that provide details on the availability of OPC UA security features, as we had no access to hardware products.

Libraries. We assessed libraries based on the availability as open-source or free (i.e., unlimited unpaid access) versions. For open-source libraries, we consulted GitHub and selected the libraries based on their popularity (e.g., number of stars), and we consulted the list at this repository[3].

3.5.2 Adoption of OPC UA features

For the artifacts considered we investigate which features of OPC UA are implemented.

Proprietary products. In Table 3.1 we summarize the adoption of OPC UA features of the 22 proprietary OPC UA products that are offered by vendors to configure their industrial devices. The table is organized into four parts: Certification by OPC Foundation, support of publish-subscribe, compatibility with Global Discovery Server, security features. Out of 22 software packages that were analyzed, four vendors rely on Codesys automation software to implement OPC UA features. None of the products considered support the publish-subscribe model. This feature, announced in the first semester of 2018, is not yet supported by products. In October 2020, Codesys software

[3]open62541: List of Open Source OPC UA Implementations (version pushed on March 10, 2020)

Table 3.1: OPC UA in proprietary products. ●/○ denotes if the product supports/not supports a feature. ◐ denotes that there are problems with feature configuration.

Vendor	Platform	OPC Cert.	Pub-Sub	GDS	Security	Trustlist	Recommended Policy
B&R	ADI OPC UA [16]	●	○	○	●	●	Not specified
Bachmann	OPC UA Client/Serv. [17]	○	○	○	◐	◐	Not specified
Beckoff	TC3 OPC UA [19]	○	○	○	◐	◐	Deprecated protocols
Beijer	iX Developer [20]	○	○	○	○	○	None
Bosch Rexroth	ctrlX CORE [24]	○	○	○	●	◐	None not supported
General Electric	iFIX [72]	○	○	●	●	◐	Basic256Sha256
Honeywell	ControlEdge Builder [95]	○[+]	○	○	○	○	None
Lenze	Easy Starter [120]	○	○	○	◐	◐	Deprecated protocols
Mitsubishi	MX Configurator-R [133]	●	○	○	●	●	None
National Instr.	InsightCM [138]	○	○	○	●	●	None
Omron	SYSMAC-SE2 [142]	●	○	○	●	●	Not specified
Panasonic	HMWIN Studio [152]	○	○	●	●	◐	Not specified
Rockwell	Factory talk linx [160]	○	○	○	●	●	Not specified
Schneider	Control Expert [57]	●	○	○	●	●	Basic256Sha256
Siemens	STEP 7 [177]	●	○	●	●	◐	Not specified
Weidmüller	u-create studio [201]	○	○	○	●	●	Basic256Sha256
Yokogawa	SMARTDAC+ [210]	○	○	○	○	○	None
Codesys based platforms							
Codesys	Codesys V3.5 [46]	○	●	○	●	◐	Not specified
ABB	Automation Builder [1]	○	○	○	●	◐	Basic256Sha256
Eaton	XSOFT-CODESYS [55]	○	○	○	●	◐	Not specified
Hitachi	HX Codesys [93]	○	○	○	●	◐	Not specified
Wago	e!cockpit [198]	●	○	●	●	◐	Not specified

[+]State of the documentation consulted during the investigation. After a preprint release of this manuscript, the documentation related to the product was updated. Now it supports security and it is certified.

released the OPC UA PubSub SL [45] extension that supports Publish-Subscribe, currently vendors do not integrate it yet. Four vendors mention compatibility with GDS servers, but it is not clear if these vendors also offer their implementation of a GDS server or provide functionality for their products to connect to third-party software.

Concerning the security features, out of 22 OPC UA servers, three vendors (Beijer, Honeywell, and Yokogawa) do not support security features. It means that deploying an OPC UA network with industrial devices from those brands will always result in an insecure deployment since the only supported security policy is None. Two vendors (Beckoff, and Lenze) support security with deprecated cryptographic primitives, making their applications de facto insecure. For the remaining 16 products that support security features, we have looked at how the user manual guides the customers through the server configuration. Specifically, we observed that two vendors (Mitsubishi and National Instruments) instruct their users to configure security None. Then, three vendors (B&R, Omron, and Panasonic) discourage the use of None, and seven vendors (Bachmann, Codesys, Hitachi, Rockwell, Siemens, Wago, Eaton) do not give recommendations on

Table 3.2: OPC UA Libraries. ●/○ denotes if the product supports/not supports a feature. ◐ denotes that there are problems with feature configuration. Security column reports if the library implements security features. Trustlist column reports if the library implements application authentication. Demo column reports if demo application supports secure connection

		OPC	Pub		Server			Client		
Name	Lang.	Cert.	Sub	GDS	Security	Trustlist	Demo App.	Security	Trustlist	Demo App.
ASNeG [15]	C++	○	○*	○	●	○	-	○*	-	-
Eclipse Milo [56]	Java	○	○	○	●	●	●	●	◐	○
Free OpcUA [68]	C++	○	○	○	-	-	-	○	-	-
LibUA [122]	C#	○	○	○	●	○	○	●	○	○
node-opcua [140]	.js	○	○*	○	●	●	◐	●	○	○
opc-ua-client [47]	C#	○	○	-	-	-	-	●	●	◐
opcua [162]	Rust	○	○	○	●	●	◐	●	●	◐
opcua [75]	Golang	○	○	○	-	-	-	●	-	-
opcua [96]	TypeScript	○	○	-	-	-	-	○	-	-
opcua4j [148]	Java	○	○	○	○	-	-	-	-	-
open62541 [149]	C	◐+	●	○	●	●	◐	●	●	◐
OpenScada UA [150]	C++	○	○	○	●	○	○	●	○	○
Python-opcua [67]	Python	○	○	○	●	○	○	●	○	○
S2OPC [163]	C	◐+	●	○	●	●	●	●	◐	◐
UA.NET [147]	C#	●	○	●	●	●	●	●	●	◐
UAexpert [194]	C++	●	○	○	-	-	-	●	●	◐

+Server certified, client not certified. *Denotes that the feature is going to be introduced in the next release

the preferred policy. Furthermore, four vendors (ABB, General Electric, Schneider, Wiedmüller) recommend a specific Security Policy (Schneider and Weidmüller use security mode SignAndEncrypt as default). Finally, one vendor (Bosch Rexroth) does not support security mode None, thereby enforcing authenticated connections. We report in the table if the certificate Trustlist is supported as required by the OPC UA specification. As we can see from the table, most vendors support it but there are several problems in the configuration procedure detailed in user manuals that make deployments vulnerable to the three attacks considered in our manuscript. We will detail the configuration issues in the following subsections.

Libraries. In Table 3.2 we report the results of our research about the adoption of OPC UA features that we conducted over 16 libraries to deploy OPC UA in industrial plants. The table is divided into four parts: support of publish-subscribe, compatibility with Global Discovery Server, server security features, client security features.

Out of 16 libraries: 11 libraries implement server features, and 15 libraries implement client features. Specifically, 10 libraries offer server and client features, 5 offer solely client features, and 1 offers solely server features. Publish-Subscribe is implemented by 2 libraries (open62541, S2OPC), also in this case, this feature is not widely adopted. At the moment of writing, the OPC Foundation official implementation (UA .NET) does not support this connection mode. GDS is implemented by 1 library (UA .NET).

With respect to security features implemented in servers, 10 out of 11 offer security features. Regarding the security features implemented in clients, 12 out of 15 implement security features. Moreover, we have investigated the correctness of security features implementation. Specifically, we looked at the availability of the Trustlist for certificate verification as described in the OPC UA standard. Among server implementations, 6 (Eclipse Milo, node-opcua, Rust opcua, open62451, S2OPC, and UA .NET) offer this feature, while the other 5 (ASNeG, LibUA, OpenScada UA, Python-opcua) do not allow the server to verify to which clients they are communicating with. Among the client implementations, 6 (Eclipse Milo, Rust opcua, open62541, UA .NET, opc-ua-client, and UAExpert) offer Trustlist functionality, while 4 libraries (LibUA, node-opcua, OpenScadaUA, Python-opcua) do not provide the feature to verify the party that they are communicating with. Regarding the remaining 2 implementations (Golang opcua, and S2OPC), we ran into issues while verifying their secure connection functionalities that prevented us from testing the availability of Trustlist features. Finally, we check the security features and verify their correct configuration in the demo applications provided in their repository.

3.5.3 Vulnerability to Rogue Server

We tested the vulnerability of 15 Client implementations from libraries, as vendors do not offer OPC UA client functionalities (apart from Bachmann). We found that all available Client demo applications are configured to connect to an endpoint with security mode None, i.e., no security. For 3 libraries (ASNeG, Free OpcUa, opcua ts) no other mode is possible. The ASNeG library will support security features starting from the next release. The 12 remaining libraries support secure connections.

For those 12 libraries, we verified their support to Trustlist for certificate management and tested it with our framework implementation. In 4 libraries (LibUA, node-opcua, OpenScada UA, and python-opcua), the Trustlist is not supported.

In 3 libraries, we had issues configuring and running the client application in a secure configuration. In Eclipse Milo, the demo client does not perform certificate validation and the documentation does not provide details about how to enable the Trustlist on although the feature is present in the source code. In S2OPC the source code features the Trustlist management, but we were not able to test it due to errors and missing details for the configuration. Finally, in Golang opcua we were not able to find information related to the Trustlist neither in the documentation, nor in issues in the repository, nor in the code, hence we assume that this feature is not supported.

In 5 clients, the Trustlist is supported, but we found two types of insecure behavior that make these clients vulnerable to Rogue Server attacks:

i) Trustlist disabled by default. In 4 libraries (Opcua Rust, UA.Net, and open62541, opc-ua-client) the demo client accepts all server certificates by default. The user can disable this option, then the libraries behave correctly w.r.t. the certificate validation procedure. While this setting is only meant to be used during development and testing, it is an additional hurdle that can lead to a seemingly secure application that is vulnerable to a Rogue Server attack.

ii) Use of Secure Channel primitives to perform certificate exchange. In

one implementation (UAexpert with GUI interface) the instructions guide the user to perform a `GetEndpoints` request to retrieve the Server certificate. The server replies to the client sending his certificate to the client. The Client prompts the error 'BadCertificateUntrusted' since the server certificate fails the security check. At this point, the client is asked to trust the certificate and re-instantiate a secure connection. This behavior is susceptible to Rogue Server attacks since the certificates are exchanged through an insecure channel. UAexpert offers the option to trust certificates before a connection to server, but this is not the default workflow in the instructions.

Overall, all 12 clients that support security features exhibit vulnerabilities that can be exploited in a Rogue Server attack. Even libraries that have handled security correctly on the server-side are lacking security features on the client-side, forcing users to lower the security properties of their OPC UA deployments.

3.5.4 Vulnerability to Rogue Client

Proprietary products. Since we do not have access to actual devices, our analysis relies on the official user manuals shipped with products. The results for the 22 analyzed products are reported in Table 3.1. The vendors Yokogawa [210], Honeywell [95] and Beijer [20] do not support any security features for OPC UA: this means that they do not offer secure communications channels to send information to clients and will not be able to perform application authentication.

For the remaining 19 companies, we have investigated if they use the Trustlist to enable only certain clients to connect to the server. All the companies implement the Trustlist for certificate verification, only 7 out of 19 correctly instruct the users to configure it. We found that 12 companies report insecure instructions to perform the certificate exchange necessary to build the Trustlist that makes them vulnerable to Rogue Client attack. In particular, we have identified two different issues with the instructions:

i) Trustlist disabled by default. The instructions by default guide the user that enables security to configure the server to accept all certificates and optionally the user can configure the Trustlist. If the server has default settings, an attacker can connect a client to the server providing an arbitrary certificate that is not verified upon trusting it. Siemens and Bachmann's products are affected by this issue.

ii) Use of Secure Channel primitives to perform certificate exchange. The issue resides in the procedure used to exchange certificates. The product affected by this issue leverages the unauthenticated `OpenSecureChannel` request to move the client certificate from client to server (instead of building the Trustlist before any connection in the network). This behavior can be leveraged by an attacker to install its Rogue Client certificate on a target server. On the server-side, the certificate is manually or automatically trusted. An operator would need to carefully check the certificate thumbprint to notice that the installed certificate is not authorized. We identified this as a common behavior in the instructions from 10 different vendor products (ABB, Beckoff, Bosch Rexroth, Codesys, Eaton, General Electric, Hitachi, Lenze, Panasonic, and Wago). All 4 products analyzed based on Codesys software propagate this insecure behavior present in the Codesys documentation. This problem may potentially threaten

more than 400 device manufacturers that rely on this Codesys software.

Libraries. We tested the vulnerability to Rogue Client attack of the 10 server libraries that offer security features. This evaluation is done with our framework implementation as described in Section 3.4. To perform our test, we started with demo applications shipped with the libraries and then verified if the library itself has additional capabilities not used in the demo to manage security features. If additional features are available, we add them to the server to test the library.

All libraries provide a demo server or a tutorial explaining how to set up a simple OPC UA server. Five demo applications offer secure and insecure endpoints, and five offer exclusively security None. For demo applications where only insecure endpoints are supported, any client can connect to the server via the security mode and policy None (i.e., without securely authenticating). Next, we tested the certificate management functionalities offered through Trustlist (mode Sign or SignAndEncrypt). We found that 6 libraries support the Trustlist of certificates, and 4 do not support it.

With our framework Rogue Client we tested the 6 demo servers provided by the implementations that support the Trustlist. Our framework tries to establish a connection in mode Sign or SignAndEncrypt where the client provides a self-signed certificate, which was not listed on the server before connecting. According to the OPC UA standard, such connections should be rejected by the server. In 3 cases the demo server rejects connection from untrusted clients (Eclipse Milo, UA.Net, s2opc). The Eclipse Milo demo server reachable online rejects unknown clients, and users are required to upload their client certificates to the server to appear in the server Trustlist and connect. Similarly, the demo servers provided by the UA.Net library and the s2opc library reject connection attempts of unknown clients. Moreover, we found that 3 demo servers show the same two types of insecure behaviors identified in the vendor section.

i) Trustlist disabled by default. Untrusted connections are allowed (by default) in 2 libraries (node-opcua, open62541), which do not enforce the use of the Trustlist to start the server. Again, we found a major flaw in the certificate management. In node-opcua there is a Boolean variable 'automaticallyAcceptUnknownCertificate' that is turned to 'true' by default when creating OPC UA server. This setting is transparent to users that are allowed to build a server without setting this variable explicitly. In the open62541 library, the user can start the server with or without the Trustlist. The Trustlist can be selected as an optional parameter to start the server from the command-line interface. When the server is started without the Trustlist, any incoming certificate is accepted, the program notifies the user is notified of this behavior.

ii) Use of Secure Channel primitives to perform certificate exchange. The Rust opcua library uses the Trustlist for certificate validation in demo applications, in our opinion the suggested procedure on certificate exchange is insecure. After a connection attempt, the client certificate is stored in the 'rejected' list of certificates from where an operator should move it to the trusted folder. An attacker can create a certificate like the real client certificate that is difficult to tell apart.

Finally, for the remaining 4 demo applications, the missing support to client certificate authentication is caused by missing features in the library altogether. In particular, we found that there are 4 libraries where the implementation of the Trustlist is missing. In the library LibUA, a function is prepared for the user with a comment to implement

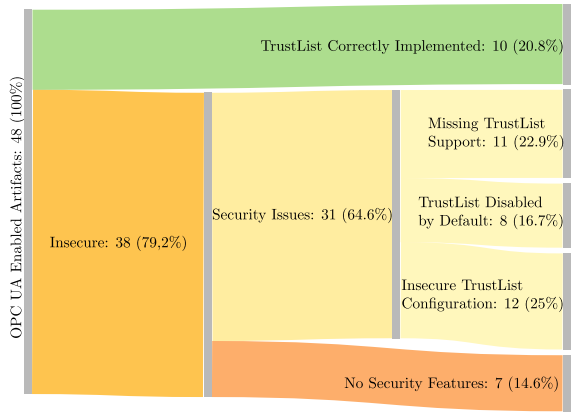


Figure 3.6: Summary of findings. It shows how the security properties for the 48 OPC UA enabled artifacts are distributed. For the artifacts that are insecure the chart details the reason of the insecure behavior. The majority of the OPC UA artifacts present security issues.

certificate authentication. Users of the ASNeG and the Python-opcua library have pointed out the missing security features in issues on GitHub. For the ASNeG library, this feature is planned for the next release [100]. Developers of the Python-opcua library in 2017 acknowledged the issue [99], but it is not available yet.

3.5.5 Vulnerability to Middleperson attack

In Section 3.4 we explained that the Middleperson attack can be performed when there are servers vulnerable to Rogue Client attacks and clients vulnerable to Rogue Server present at the same time in the network. Combinations of aforementioned OPC UA servers and clients that are vulnerable respectively to Rogue Client and Rogue Server attacks would make a deployment that is vulnerable to Middleperson attacks. In particular, we have identified that 19 servers' artifacts out of 29 (65%) that support security features are vulnerable to the Rogue Client attack (4 artifacts due to missing Trustlist features and 15 due to insecure instructions Trustlist) and 12 clients out of 12 (100%) that support security are vulnerable to Rogue Server attack (7 artifacts due to missing Trustlist features and 5 due to insecure instructions to configure Trustlist). As we can see there is a non-negligible risk to deploy an insecure OPC UA network where an attacker can operate as Middleperson.

3.5.6 Summary of Findings

We considered a total of 48 OPC UA enabled artifacts: 22 products from vendors and 16 libraries (of which 11 provide OPC UA servers, 15 provide OPC UA clients). Figure 3.6 reports a summary of our findings that we detail in the following.

Seven artifacts do not support security features at all (14.6%). Among the 41 remaining artifacts that support security features, we found that 31 artifacts (64.6% of the 48 OPC UA artifacts) show issues or errors in the Trustlist management that enable Rogue Client, Rogue Server, and Middleperson attacks. The other 10 artifacts (20.8%

out of 48) correctly implement the Trustlist management and instruct users about its configuration, and thus they are not vulnerable to the Rogue Client, Rogue Server, and Middleperson attacks. The 31 artifacts that show security issues with the configuration of the Trustlist can be classified into the following three categories:

- **Missing Support for Trustlist.** 11 artifacts do not implement Trustlist (or do not provide instructions about its configuration) management although they implement OPC UA security features, that is they offer functionality for signing and encryption but not for the validation of certificates. This makes their deployments always vulnerable to Rogue Client, Rogue Server, Middleperson attacks.
- **Trustlist disabled by default.**

  In 8 artifacts the Trustlist is disabled by default. This behavior puts OPC UA deployments at risk as applications will accept any incoming certificate, making them vulnerable to the three considered attacks.
- **Certificate exchange through Secure Channel primitives.** In 12 artifacts the instructions guide the users to use unauthenticated Secure Channel primitives to perform certificate exchange. The user is guided to initiate a connection such that the connecting applications send their certificates to each other. Then the certificates are manually trusted for each device. Since the certificates are sent via an insecure channel an adversary can leverage this behavior to mount an attack. The OPC UA standard allows this behavior [143] assuming that only trained personnel are authorized to trust incoming certificates exchanged through insecure channels. In our opinion, this recommendation is flawed as insecure channels allow simple manipulation of the certificates before acceptance by the administrator (via Middleperson attacks), and humans have been shown to be bad at detecting manipulated certificates [53, 42].

Adoption of features. in our analysis of OPC UA features, we found that the client-server model is available in all tested products while the publisher-subscriber model is supported by 3 artifacts. Moreover, the features offered by the GDS to manage certificates are supported by 5 artifacts. Security features are widely adopted, but they require the correct distribution of certificates to deploy a secure network. As we found, this is not always the case in OPC UA products as many of them do not implement the Trustlist for certificates or do have issues in the configuration procedure.

Our investigation is the first about security of OPC UA artifacts. Our findings and considerations are consistent with prior work in usable security of cryptographic APIs [4, 83], which showed that lack of comprehensive documentation and demo applications will result in poor security configurations implemented in the final deployments.

3.6 Countermeasures

Our work showed general missing support and incompleteness of OPC UA security features in artifacts that make the certificate management often not possible or not required by default. Moreover, we discovered some insecure behaviors allowed by the

OPC UA standard. In this section, we discuss countermeasures to prevent vulnerabilities in OPC UA deployments and achieve an initial secure key distribution.

3.6.1 Certificate exchange via insecure channels

As explained in Section 3.3, initial key distribution for air-gapped ICS is a non-trivial problem as devices cannot be shipped with root CAs installed. OPC UA Global Discovery Servers with Certificate Manager would be an alternative to manual certificate exchange, but as we showed in Section 3.5 this feature is not widely implemented and, in any case, the GDS certificates will not be shipped together with devices newly introduced into the system. That implies that bootstrapping the security in an OPC UA system critically relies on manual pre-distribution of certificates even when GDS is deployed.

Currently the OPC UA standard does not provide effective solutions to overcome this challenge, instead (as explained in Section 3.5) it guides the users to exchange certificates through insecure channels. Given the technological challenges posed by ICS networks, we suggest to i) update the standard (and manuals) to instruct users to rely on out-of-band secure channels for initial certificate distribution, ii) enforce Trustlist population before use of OPC UA secure channels and not through unsecured primitives as found commonly in the consulted user manuals and the standard itself.

Out-of-band certificate exchange can be achieved with different solutions e.g., through secure protocols such as SSH, or through physical solutions such as USB sticks and QR codes [128]. Each of those viable solutions has different trade-offs in terms of usability and security but in any case, offer better guarantees than exchanging certificates through unsecured primitives. Of course, exchanging OPC UA certificates through SSH requires that the communicating devices have already shared SSH secrets. USB sticks are a widely supported solution to distribute certificates in the industrial environment, as a drawback the users are required to physically move around the plant to distribute the certificates to the devices that are expected to communicate with each other. Finally, QR codes represent another potentially easy to use and deploy solution to distribute certificates in the industrial environment [128]. In particular, industrial devices could be shipped with a private public key pair, the public key can be printed on a QR code sticker and physically applied to the device. During the installation process the operator scans the QR code and installs the certificate in the Trustlist of the authorized devices in the industrial environment. As a drawback of this method, an attacker could deploy a malicious device in the industrial plant while performing a supply chain attack and replace the QR code sticker with his own certificate on legitimate hardware.

3.6.2 Missing support for Trustlist

End users should only use products and libraries that implement the Trustlist management as described in the OPC UA standard. In particular, products certified by the OPC can be expected to support this feature. Artifacts that do not provide this feature should instead implement it in order to allow communicating parties to trust each other upon connection.

3.6.3 Trustlist disabled by default

End users should enable the Trustlist if disabled upon OPC UA network configuration. Vendors recommend enabling the Trustlist by default, and thus making it mandatory to populate the Trustlist upon the creation of secure channels. Enabling the Trustlist functionality will require the user to perform additional steps to set up a new device on the OPC UA network but those steps are of crucial importance for the security of the ICS.

3.7 Related Work: Industrial Protocol Security and Usable Security

3.7.1 Security of open source OPC UA libraries

Muhlbauer et al. [136] study the security of four popular open-source libraries (UA .Net Standard, open62541, node-opcua, Python-opcua) which we also inspect in this chapter. The analysis focuses on five aspects: dependencies, timeouts, supported Security Policies, message processing, and randomness. The authors identify two vulnerabilities in the implementations: missing upper time limits in Python-opcua making it vulnerable to DoS attacks and missing packet type checks in node-opcua. The issues related to the Trustlist configuration of the libraries investigated our work were not identified.

Neu et al. [139] and Polge et al. [157] showed the feasibility of DoS attacks by a Rogue Client. They suggest network traffic anomaly detection as a countermeasure. Polge et al. [157] implemented a Middleperson attack using Eclipse Milo. The attacked OPC UA applications are using Security Policy None or Aes128-Sha256-RsaOaep. They report that if username and password are sent as part of the ActivateSessionRequest message the password is encrypted even in Security Mode None and can therefore not be recovered (in contrast to our findings). The encryption of the password in Mode None is optional in the Specification [144], this design choice was taken by [73]. Encrypting credentials with untrusted certificates is susceptible to Rogue Server attacks. Based on our findings, a Middleperson attack that recovers plain-text credentials is possible. Therefore, a Middleperson attacker can provide their own certificate, and the user credentials are encrypted using the public key associated with the attacker's certificate.

3.7.2 Secure exchange of certificates

The security of the OPC UA protocol relies on certificates that authenticate each application. Prior publications proposed techniques to establish the initial root of trust. In [105], a PKI is implemented that offers functionality to sign certificates or verify certificates using the Online Certificate Status Protocol. Meier et al. [130] propose to connect a new OPC UA application to a physical device with certificate manager functionality to install a certificate and the Trustlist before connecting the application to the network.

3.7.3 Usability issues leading to insecure systems

Usability of security features is an active research topic that relates to the usability of OPC UA security features. Several studies point out that the integration of security features into programs and systems has to be facilitated [83].

Acar et al. [4] compare the usability of five Python cryptographic libraries. They find that APIs which offer fewer options lead to better security results. In addition, the documentation of the library and the availability of example code had a stronger influence on the successful completion of the task than the experience of the participant.

Krombholz et al. [115] conducted a usability study on the configuration of HTTPS. In the study 28 system administrators were asked to configure a web server with HTTPS. They found that participants struggled to find reliable resources to learn the process, a large number of configuration options were difficult to comprehend, and the default configuration only offered weak security. Additionally, the security benefits which a protocol such as HTTPS offers are misunderstood or underestimated. Based on misconceptions some administrators decide against using secure options [114, 59].

3.8 Conclusions

Usable security of libraries is an active research area [4, 83, 115, 114] and showed how important features and documentation are for secure deployments. In this work, we have systematically investigated practical challenges faced to use OPC UA securely. To this end, we introduced a security assessment methodology—our assessment considers three attacks that can target OPC UA deployments, Rogue Server, Rogue Client, and Middleperson attacks.

We systematically address three research questions: **RQ3.1** What are practical challenges for the correct use of OPC UA security features? **RQ3.2** Are OPC UA security features correctly implemented by the vendors and products? **RQ3.3** What are the consequences of breaking OPC UA security features?

To address **RQ3.1**, we conducted the first systematical survey of 48 OPC UA artifacts provided by vendors or open source. Our survey investigates the availability of OPC UA security, publish-subscribe, and Global Discovery Server functionalities. We showed that publish-subscribe and Global Discovery Server are not widely adopted. Furthermore, we showed that 7 OPC UA artifacts do not support security features of the protocol.

To address **RQ3.2**, we proposed a framework to investigate the identified security issues. With our framework, we show that the identified issues make OPC UA artifacts vulnerable to the considered attacks. We analyzed 48 OPC UA artifacts, 41 of those artifacts support security features. We found that 31 out 41 artifacts present three recurring pitfalls in the Trustlist management to establish the initial root of trust.

To address **RQ3.3**, we designed, implemented, and demonstrated three types of attacks. The attacks allow the attacker to steal user credentials exchanged between victims, eavesdrop on process information, manipulate the physical process through sensor values and actuator commands, and prevent the detection of anomalies.

Our findings demonstrate major security flaws in OPC UA artifacts that threaten the

OPC UA security guarantees. Those results imply that there are a significant number of OPC UA deployments in industry that either do not provide full security guarantees or rely on the absence of the attacker at configuration time for new devices to bootstrap their authentication process. We believe that our proposed systematic approach is useful to increase awareness among users, developers and companies about the threats that can be produced by the three identified pitfalls in the initial key establishment and Trustlist configuration. As a complementary result, our POC tool can be used as a tool to probe for erroneous configurations and improve security in OPC UA deployments.

In this chapter, we demonstrated how achieving secure communication in a CPS remains an open challenge, despite secure-by-design protocols being proposed. In the next chapter, we look into the effects on the CPS process caused by message manipulation or injection.

4
Robotic Vehicles Security

Sensor Deprivation Attacks for Stealthy CPS Manipulation

4.1 Introduction

In modern society, Robotic Vehicles (RVs) such as Unmanned Aerial Vehicles (UAVs) are deployed to accomplish a number of tasks. UAVs have been used to deliver medicines [179] and tested for life-saving deliveries [63]. UAVs are used for consumers' good delivery [203, 199]. Given their importance, security concerns were raised. Recently, a number of works studied the UAV's security and showed critical issues affecting popular flight stacks [180, 167, 103, 102, 171].

In UAVs, sensors are generally continuously queried by the Micro Controller Unit (MCU) for the latest sensor readings. Sensors are configured to update their readings with fixed sampling rates (higher than the MCU query rates). As sensors are in dedicated chips, they transmit the data to the MCU via the local bus system or analog signals. So far, it is unclear *what would happen to the drone control if the attacker can reduce the sensor update rate, or prevent any sensor data from being delivered?*

In this chapter, we introduce Sensor Deprivation Attacks (SDAs) and investigate their impact on modern UAVs. SDAs are a new class of attacks that aim to manipulate the sensor update rate (in the most extreme case disabling updates) to prevent the correct sensor reading from reaching the controller. In contrast to a sensor spoofing attack, in an SDA, the intruder does not need to craft and inject malicious traffic into the network actively. The SDA only requires intermittent access and limited resources.

We first demonstrate the feasibility of the device reconfiguration via an unauthenticated bus. Then we formalize the attacks and evaluate their impact on Commercial off-the-shelf (COTS) flight controllers and emulation environments. We then investigate practical realizations of these abstract attacks. In particular, we propose two strategies to launch SDAs in the system by reconfiguring sensors, namely *Sensor Suspend attacks* and *Sensor Sampling Frequency attacks*. Both attacks leverage the fact that sensor chips allow configuration via unauthenticated serial APIs. These APIs allow changing register values corresponding to a specific operational mode of the IMU. Thus, an attacker changes the sensor behavior with single injected messages on the bus. Finally, we propose a Reinforcement Learning attack synthesis methodology to strategically fly the drone via intermittent SDAs.

Our experimental results show that SDAs induce severe consequences on UAV stability, quickly leading the drone to deviate or crash from the planned trajectory. Proposed SDAs impact the controller performance without actively modifying it (i.e., without causing any abnormal operation into the flight controller code, which an Anomaly Detector might monitor). Finally, we show that the proposed SDAs can be controlled to fly the UAV to an attacker's desired location.

The contributions of the chapter are:

- We propose and formalize Sensor Deprivation Attacks (SDAs) and experimentally investigate the impact of SDAs on dynamic control systems.
- We investigate practical options to realize SDAs on integrated UAV platforms and propose a novel approach leveraging the reconfiguration of sensors via unau-

thenticated buses (to suspend them or reduce their sampling frequency). We demonstrate the practical feasibility of the message injection on a I^2C bus and the impact of this attack on five different hardware platforms. We find severe effects on two different control software stacks.

- We propose an attack synthesis methodology to optimize SDAs for deterministic manipulation of drone behavior. This is the first time that drone manipulations do not only lead to uncontrolled crashes, but deterministic control of the drone without active sensor spoofing.

4.2 Background

4.2.1 Serial Buses for Sensor Communication

In the UAV architecture, sensors communicate via serial buses to the microcontroller. Examples of these serial buses are SPI, I^2C, CAN, and UART. These serial bus protocols are not authenticated. The same unauthenticated protocols are also used for sensor configuration, done by the microcontroller via serial APIs. The application running on the microcontroller changes the configuration registers on the sensor in order to configure the sensors to satisfy the application's real-time control requirements.

For the purpose of this chapter, we introduce the readers to the I^2C protocol, which is commonly used for serial device communication. The I^2C protocol uses a two-line bus to communicate among devices, a clock signal, and a data line. All communication is coordinated by a central controller, typically the main MCU of the system. All other bus members listen to the data line and respond to the specific controllers' requests. The main controller mandates when the devices communicate via the clock signal.

4.2.2 Serial Bus attacks via IEMI

Prior work demonstrated Intentional Electromagnetic Interference-based (IEMI) attacks to disrupt serial bus communication.

Selvaraj et al. [173], demonstrated how via IEMI, analog sensor transmission can be manipulated to produce arbitrary sensor and actuator values, moreover, they show the induction of bit flips in serial communications. Dayanıklı et al. [52] demonstrated false actuation attacks via IEMI in the context of PWM actuators for UAVs. Dayanıklı et al. [51], demonstrated bidirectional bit flips via IEMI in serial protocols such as UART and I2C. Zhang et al. [212], demonstrated the effects of IEMI on differential signals, showing that such systems are also vulnerable to IEMI induction. The work by Jang et al. [102] shows the impact of IEMI on disrupting the sensor communication context of UAVs. Xie et al. [207], demonstrated UART bit-level bus manipulation via IEMI injection and proposed a countermeasure to mitigate such attacks.

The use of IEMI for bus manipulations has been extensively studied and demonstrated successful in disrupting serial communication. Current UAV architectures remain vulnerable to such manipulations. In our work, we rely on this attack vector to mount our proposed methodology. In our work, instead of continuously manipulating the sensor readings, we propose to use IEMI to achieve victim sensor reconfiguration.

4.2.3 Attacks on UAVs

Nassi et al. [137] systematized the state-of-the-art UAVs' attacks and defenses. We summarize some of the attacks investigated by prior works. Son et al. [180] demonstrated how sound injection attacks can affect gyroscopic sensors and affect drone controllability. While Jeong et al. [103] investigated how to remediate sound injection attacks. Sathaye et al. [167] experimentally investigated sensor spoofing attacks on drones and demonstrated that the takeover and control of a drone are challenging and require real-time spoofing signal manipulation. Dayanıklı et al. [52], investigated the use of false PWM actuation commands via IEMI to adversarially fly a victim UAV plane. Jang et al. [102] investigated the effect of IMU signal corruption by actively blocking serial communication channels or performing electromagnetic emission investigations to prevent sensor signals from reaching the controller.

4.2.4 Anomaly Detection for UAVs

Open-source autopilots such as Ardupilot deploy the EKF failsafe [12] mechanism, it monitors the variance of the Extended Kalman filter innovation (residuals) and applies threshold-based triggers to classify an anomaly on the drone. According to the specification, the failsafe is triggered when the EKF innovation exceeds the threshold for more than 1 second. As a detection response, the drone will switch to land mode. Similarly, prior work in the security for Robotic Vehicles proposed in-vehicle anomaly detection. Quinonez et al. [158] proposed to detect anomalies by applying an additional Extended Kalman Filter monitor and the CUSUM change detection algorithm on the filter residuals. Choi et al. [44] proposed to use control invariants derived via system identification to detect anomalies in the target system.

Finally, more recently, Khan et al. [107] proposed to detect attacks affecting the control system by monitoring the microcontroller's low-level peripheral bus registers to identify any abnormal access patterns, and by moving the Kalman filter estimation outside the RTOS.

In this chapter, we consider the built-in failsafe mechanism from Ardupilot and the M2MON.

4.3 Motivation and Assumptions

4.3.1 System and Attacker model

System Model. We consider an UAV architecture as described in Chapter 2, equipped with a flight controller (Figure 2.3) running state-of-the-art autopilot software. The flight controller comprises a Micro Controller Unit (MCU) which runs the control software and is connected to various peripheral devices such as sensors, actuators, and radio modules. All these components communicate via buses such as I^2C, with the MCU acting as the sole bus coordinator, and the other devices reply to its queries.

The MCU runs the control software relying on a Real-Time Operating System (RTOS) offering scheduling functionalities essential to handle the tasks required to control the UAV. The high-level system architecture of the control software includes

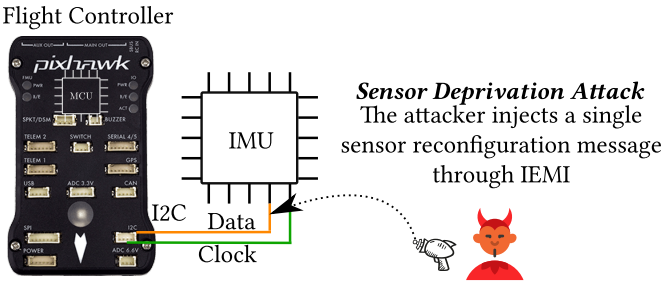


Figure 4.1: Attacker Model Example—The attacker deploys an IEMI attack and sends a command to the victim IMU sensor to change its configuration.

Table 4.1: Attacker model comparison. Compared to prior work attacks against UAV sensors, our attack does not require continuous signal injection or active spoofing of the sensor value. Instead, our attacker only requires injecting a single reconfiguration message on the bus.

	GPS spoofing[167]	Sensor spoofing [102, 180, 103]	Our SDA
Continuously inject signal	●	●	○
Actively spoof sensor	●	○	○
Single command injection	○	○	●
Deterministic control	●	○	●

the drivers to interface with the external sensors (which are periodically queried to pull the fresh readings), Kalman Filter to produce state estimation and update via sensor fusion, controllers (e.g., Attitude, Position, and Motor controllers).

A monitoring system like M2MON [107] may be in place to detect attacks on the UAV by monitoring the Kalman filter and the serial register. In its most basic form, the monitor will check for unexpected large differences in consecutive sensor readings (e.g., bad data detection [32]).

Attacker Goal. An attacker aims to deviate a drone from its expected trajectory or crash it, without being detected by the monitoring system.

Attacker Model. We assume the attacker can inject messages or symbols on the serial unauthenticated communication channels (e.g., I^2C bus) in the drone (Figure 4.1). Two attack strategies were demonstrated in prior work for bus message injection. Injection can occur via intentional electromagnetic emission (IEMI) on serial buses (see Section 4.2.2). Injection can occur via a compromised device connected to the shared bus [38, 60].

Positioning against related work. Table 4.1, summarizes the differences between our proposed attack and prior proposed sensor spoofing or false data injection attacks (see Section 4.2.3). Our proposed attack does not require the attacker to actively and continuously generate attack traffic to substitute/disturb the legitimate traffic (e.g., as in a GPS spoofing). Differently from prior work attacks, the traffic leading to control destabilization is generated from the legitimate source without breaking the code execution or corrupting the communication channel.

4.3.2 Attack Idea

In a nutshell, the main idea of the attack is twofold: i) the attacker leverages unauthenticated bus access to manipulate or misconfigure devices on the bus to influence sensor measurements stealthily, ii) using this capability, the attacker forces the victim controller to perform wrong control decisions, leading the drone on the wrong path or crashing it. There are two main challenges for this: i) it is not clear from prior work what kind of manipulations are possible for the attacker in this scenario without raising errors or alarms at the controller, ii) depending on the impact the attacker can have on the sensors, deterministically controlling the effects on the drone behavior will be challenging.

As we will show in Section 4.4, in this scenario, the attacker will be able to reconfigure selected sensors on the drone (see Section 4.2.1), leveraging the fact that sensor configuration is not authenticated or verified at runtime. This reconfiguration will cause selected sensor readings to be denied or delayed at the MCU. Our attack requires only single messages to reach the sensor. In contrast, continuous spoofing or manipulation of bus traffic is much less reliable and could trigger detection mechanisms (see Section 4.5).

In Section 4.5, we formalize the attack from a control theory perspective and identify the possible induced behavior on the control algorithm. In Section 4.6 we identify possible sensor (re)configurations that reduce the sensor sampling frequency and turn off the sensor completely and induce wrong control actions. We investigate this in detail on drones.

To capture the intuition of this novel generalized type of attack targeting configuration of sensors, we introduce the term *Sensor Deprivation Attacks* in this chapter. We will define this more formally in Section 4.5.

4.3.3 Potential of Sensor Deprivation Attacks

We now present a potential application of sensor deprivation attacks on drones and its possible effects. Figure 4.2 shows an example: while a drone is following a specific mission (e.g. moving along waypoints), the attacker is launching targeted attacks to suppress specific (or all) new sensor readings (e.g. through sensor reconfiguration). Such attacks cause the drone controller to operate on outdated sensor readings and keep its dynamics unaltered with constant actuation – which causes the drone to go on a wrong trajectory. When the attack stops, the drone will compute the correct state estimation and attempt to re-stabilize the frame. Figure 4.2, reports an example where the drone crashes on the ground while attempting to recover from the SDA. We note that in the context of Industrial Control Systems, a related concept of Stale Data attacks [116] was previously introduced to manipulate industrial controllers to perform control decisions based on outdated previous sensor readings by suppressing newer sensor values in traffic.

We claim that attacks on Cyber-Physical Systems in which the attacker denies timely sensor updates represent an interesting class of under-explored attacks in three main aspects: a) What are options for the attacker to lead to the Sensor Deprivation Attacks, b) What are the effects of such Sensor Deprivation Attacks on the input provided to the victim controller, c) What are the effects on the control decisions taken by the victims? We note that for all aspects, alternatives to the setting described in as Stale data attacks

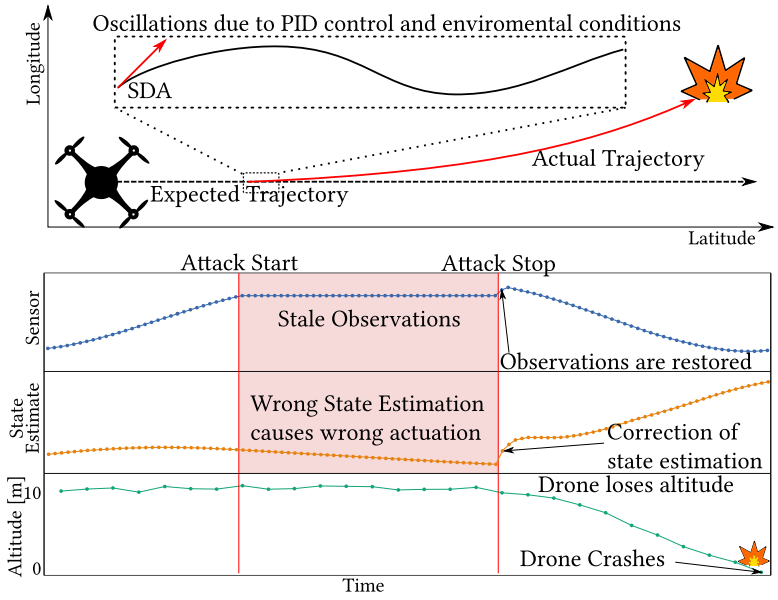


Figure 4.2: Motivating example, a drone flies on a straight trajectory. The flight is influenced by sensor and environmental noise which results in continuous correction applied to the flight. An attacker starts a Sensor Deprivation Attack, targeting one of the drone sensors. Consequently, the attitude estimation is compromised, causing wrong actuation commands which will result in the drone deviation or crash.

in [116] are possible. For example, sensor updates could be denied by manipulating the sensors to give updates less frequently, or by disabling sensors temporarily. In addition, effects on the controller could range from the re-use of old values to (accidental) use of error values as sensor readings, busy waiting in software for sensor replies, or the use of random values instead.

4.4 Drone Bus Manipulation

As summarized in Section 4.2.2, prior work demonstrated the use of IEMI to manipulate UAVs' bus communications, for this reason, we assume that it is possible in general. In this section, we explore the feasibility of injecting maliciously crafted commands into a shared bus. Particularly, we want to address the following challenges. **RQ4.1** Can a compromised peripheral also maliciously act as a controller? **RQ4.2** Which interactions with the bus will be stealthy, i.e., will not lead to alarms or errors at the legitimate bus controller?

To answer these questions and explore the feasibility of the bus manipulation, we design and demonstrate a two-step bus manipulation on an I^2C bus (see Section 4.2.1) testbed.

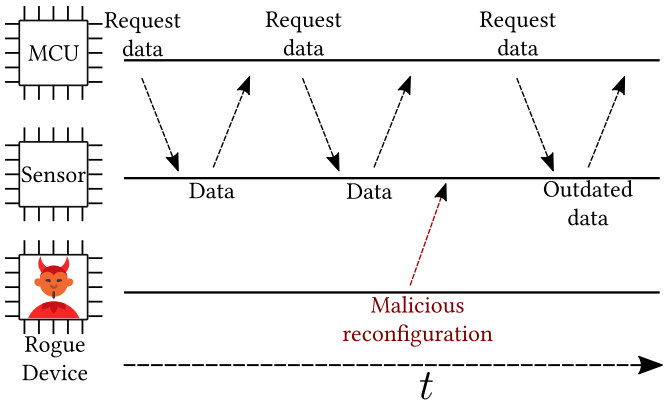


Figure 4.3: Attacker Model (communication). An attacker who gained access to a shared bus can inject a reconfiguration message into the victim sensor as the bus is not authenticated. The attack is transparent to the victim MCU, which will start receiving outdated or erroneous data from the sensor.

4.4.1 Bus Manipulation via Shared Bus

Experimental Setup. We set up a testbed with a main controller (Raspberry Pi3) that connects two peripherals through a shared bus (I^2C), an IMU (MPU6050), and a third-party peripheral (Raspberry Pi4). The main controller constantly reads the gyroscope data from the IMU.

The attacker-controlled peripheral then unilaterally reconfigures its I^2C interface from peripheral to controller mode (competing with the existing controller), using Adafruit's CircuitPython library. A 50-line Python script contains the initial sensor setup, the periodic sensor request, and the attacker's passive monitor and command injection functions.

Challenges with bus manipulation. We started with experiments on bus collisions, which occur if two controllers communicate simultaneously. Collisions can inject unexpected values into target registers and result in unpredictable behavior. We found experimentally that once the controller detects the collision, it stops transmitting. Moreover, I^2C uses pull-up resistors in both lines to always guarantee a logical '1', which it makes impossible to have full control of the data line, as only '1' can be turned into '0' and not vice-versa. This makes message manipulation challenging.

Fortunately, we found that the clock signal is generated only when the main controller sends a request on the bus. This enables an attacker to leverage malicious command injection.

Eavesdropping. As a consequence of our initial finding, when the bus is available, any other device could generate the clock signal and maliciously impersonate the legitimate controller on the bus (see Figure 4.3). An attacker can listen to the shared bus's reading patterns and understand when crafted malicious commands can be injected to avoid collisions.

Injections. Once the bus is available, the attacker has a window before the next communication with the legitimate controller occurs. Within this time window, our attacker sends requests to the target peripheral on the bus to change its configuration.

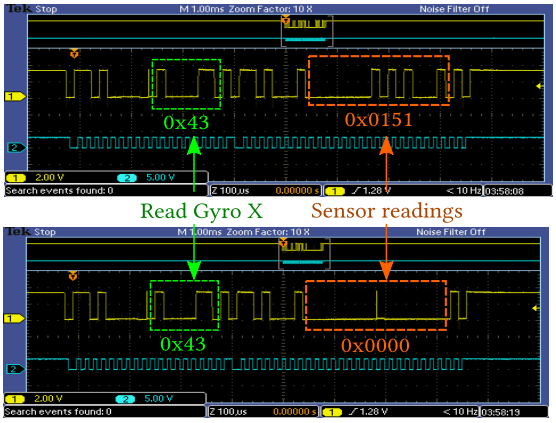


Figure 4.4: Sensor reconfiguration, oscilloscope visualization: Reading Gyro X on the shared bus. Top: Readings before the malicious command injection. Bottom: Readings after the malicious command injection, no data are transmitted due to the device reconfiguration.

Command Injections Demonstration. In Figure 4.4, we show the I^2C reading command on an oscilloscope before and after a successful IMU reconfiguration (in this case, reconfiguration to sensor standby). The figure depicts the read gyro X command `0x43` (in green) and the sensor response (in orange), `0x0151` before and `0x0000` after the attack. More specifically, the sensor stops answering the controller requests after reconfiguring the sensor.

Conclusions on Bus Manipulations. We practically realize and demonstrate the reconfiguration command injection on a I^2C bus testbed. Regarding **RQ4.1**, our results show that it is possible to have a malicious device acting as a controller on the shared bus. Regarding **RQ4.2**, directly manipulating the sensor readings will likely lead to collisions on the bus, which will be detected. Moreover, not every manipulation is possible. For this reason, we rely on stealthy single message injections at appropriate times.

Generalizability to other protocols. We note that the same approach applies to other serial protocols (like CAN). Independently of the bus protocol used, two factors make the reconfiguration possible, i) the use of unauthenticated buses ii) the reconfiguration feature on the target sensor.

4.5 Sensor Deprivation Attacks

In this section, we introduce SDAs. We start by providing the attack definition and formalizing it. Next, we explore the impact of the proposed attacks on the control algorithms in an emulated environment. This emulation will serve as the motivation to further explore the feasibility of the proposed attacks in real COTS hardware (Section 4.6). Specifically, we want to address the following question **RQ4.3**: *What are possible behaviors resulting from Sensor Deprivation Attacks and how can they be formalized?* and **RQ4.4**: *What is the impact of the proposed Sensor Deprivation Attacks on UAVs?*

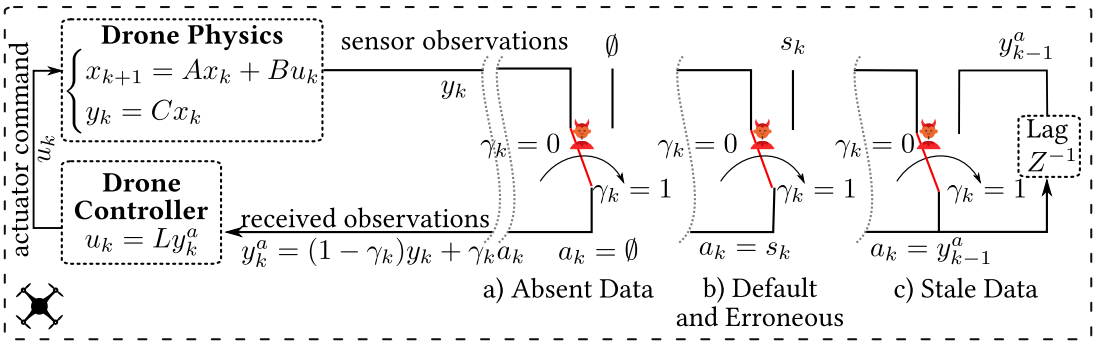


Figure 4.5: Sensor Deprivation Attack abstraction. An attacker causes Sensor Deprivation Attack between the sensor and the controller. Different behaviors can be observed at the controller, a) Absent data, where no data is received b) Default data where s_k is constant and Erroneous data where $s_k \sim \mathcal{N}(\mu, \sigma^2)$, c) Stale data where last observation is received.

4.5.1 Attack Intuition

We use the linear system representing the interaction between the controller and the physical process [170] to introduce the SDA concept. Consider the system depicted in Figure 4.5. Based on the sensor readings, the controller computes the actuator signal to control the physical process.

Intuitively, our SDAs are the attacker's actions to 'interrupt' the feedback connection from the physical process to the controller (in Figure 4.5, the 'switch' symbol corresponds to the attacker's action). So, after the attacker influences the system, an attacked set of observations reach the controller.

In control theory, prior work proposed control strategies for lossy sensor/control packets [170, 169, 178]; in particular, the missing observations are stochastically modeled. In contrast, we assume that a strategic attacker deterministically triggers the lossy communication to destabilize the process.

We identify four possible behaviors that can be modeled in the case of sensors' deprivation, and their resulting missing observations (i.e., missing sensor readings), inspired by prior work in control theory [170].

- ***Stale data.*** In this setup, the sensor stops updating, and the last observation is reused by the controller until a new observation is received. Krotofil et al. introduced this setup in an ICS scenario [116].

- ***Default value data.*** In this setup, the value that reaches the controller is incorrect and its value is constant. This situation might occur when the sensor is in an error state.

- ***Erroneous data.*** In this setup, the value that reaches the controller follows a certain statistical distribution (noise). This might occur when some disturbances in the sensor prevent correct sensing (e.g., acoustic signal interfering with the sensor [180]) or disturbances in the communication channel [102])

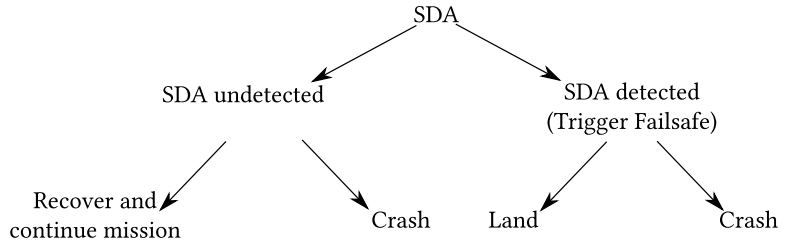


Figure 4.6: Possible behaviors after a Sensor Deprivation Attack

- ***Absent data.*** No sensor values are received at the controller. This situation might occur when the sensor is disabled.

In the case of *absent data* two possibilities for the controller were proposed by Schenato et al. [169].

- ***Zero-input strategy.*** Stop actuating the system.
- ***Hold-input strategy.*** Keep actuating based on the last observed value (or keep the last control action), as described by Krotofil et al. [116].

In the case of *default data*, *erroneous data*, and *stale data* from the sensor, the controller will continue actuating based on the received data.

4.5.2 Attack Formalization

We now formalize the proposed attacks Sensor Deprivation Attacks to address RQ4.1. We consider a linear system, Equation 4.1.

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{4.1}$$

where A, B, and C are matrices that describe the coefficient physical model. The system operates in a discrete timer k. x_t denotes the current system's state, y_k are the sensor readings, and u_k denotes the actuation command that is computed based on the current sensor readings. In the case of SDAs, the observations that reach the controller are affected by the attacker (y_k^a) which can arbitrarily decide to start the attack on the system ($\gamma_k \in \{0, 1\}$).

In the case of Absent data, when an attack starts, no sensor data is received at the controller, and the sensor observation model becomes $y_k^a = (1 - \gamma_k)y_k + \gamma_k\emptyset$. In the case of Default value data, the observation model becomes $y_k^a = (1 - \gamma_k)y_k + \gamma_k s_k$ where s_k is constant, while in the case of erroneous data $s_k \sim \mathcal{N}(\mu, \sigma^2)$. Finally, in the case of stale data, the observation model becomes $y_k^a = (1 - \gamma_k)y_k + \gamma_k y_{k-1}^a$ where y_{k-1}^a is the last sampled value by the sensor before the attacks start.

4.5.3 Sensor Deprivation Attack Emulation

We now evaluate the effect of SDAs on the Ardupilot [10] autopilot system (version 4.3.3 from January 2023), running in the Software In The Loop (SITL) simulator. SITL

is a fully functional environment to run Ardupilot software in a simulated physical environment for testing and verification purposes.

All the Ardupilot features are available within the SITL framework, for this reason, it is well suited to test the impact of SDAs on the control algorithms without safety hazards. To emulate the effects of the SDAs we target the virtual sensor driver for the IMU (`AP_inertial_sensor_SITL.cpp`). In turn, we emulate the Stale data by storing the last observation and re-transmitting it to the attitude controller, the Default data by transmitting a constant value, the Erroneous data by transmitting random data, and the Absent data by skipping the data transmission. We create a straight line mission, with a desired altitude of 50 meters and fly it 55 times, 11 times without attacks to create a baseline trajectory, and 11 times launching each SDA for the duration of 1 second to evaluate the impact of the attack. The SDAs is launched when the drone reaches the desired altitude and follows the desired straight trajectory. We evaluate the impact of the attacks in terms of deviation D (Equation 4.2) as the instantaneous mean squared difference between the attack trajectory A and the reference trajectory R. The distance is computed over the three-dimensional space axis, x (longitude displacement), y (latitude displacement), and z (altitude displacement).

$$D_{t(x,y,z)} = \sqrt{(A_{t(x,y,z)} - R_{t(x,y,z)})^2} \tag{4.2}$$

In particular, we consider Maximum deviations (Equation 4.3) induced by the attack (i.e., between the events *attack start* and the *crash*, *failsafe* or *landing* event timestamps).

$$\max \quad D_{t(x,y,z)} \quad t \in \{\text{attack start, failsafe/crash}\} \tag{4.3}$$

Moreover, we are interested in evaluating the impact of the attacks on the Ardupilot failsafe mechanism (see Section 4.2.4). For this reason, we measure how often the attack triggers the failsafe, how often the drone crashes, how often the drone completes the mission, and how often it lands after triggering the failsafe (see Figure 4.6). For the attacks that trigger the failsafe or crash the drone, we compute respectively the average Time to Detect (TtD) and Time to Crash (TtC), we note that with the Ardupilot default failsafe the minimum Time to Detect is 1 second (see Section 4.2.4).

$$TtD = T_{failsafe} - T_{attackStart} \tag{4.4}$$

$$TtC = T_{crash} - T_{attackStart} \tag{4.5}$$

4.5.3.1 Emulation Results

Table 4.2 and Table 4.3, report the summary of the emulation carried out over the four SDAs. Figure 4.7 and Figure 4.8, report the distribution of the Time to Detect and Time to Crash. In our evaluation, we measure the deviation induced by the SDA before the failsafe is triggered.

Stale Data. In the case of Stale Data attacks, the attack triggers the failsafe 100% of the times, which successfully lands the drone 100% of the times. Concerning the time to

Table 4.2: Emulation of Sensor Deprivation Attacks with Ardupilot SITL. Behavior characterization of Ardupilot flight control after launching Sensor Deprivation Attacks for 1 second. Each attack is repeated 11 times.

		SDA behavior characterization				TtD		TtC	
		Undetected		Detected					
	%failsafe	%crash	%completed	%crash	%landed	μ(s)	σ	μ(s)	σ
Stale	100.00	0.00	0.00	0.00	100.00	4.99	1.80	N.A.	N.A.
Default	100.00	0.00	0.00	100.00	0.00	1.22	0.00	4.56	0.00
Erroneous	27.27	45.46	27.27	27.27	0.00	1.76	0.47	5.66	2.96
Absent	0.00	100.00	0.00	0.00	0.00	N.A.	N.A.	4.22	0.01

Table 4.3: Emulation of Sensor Deprivation Attacks with Ardupilot SITL. Sensor Deprivation Attacks induced trajectory deviation. Attack duration is 1 second. Each attack is repeated 11 times.

	Maximum Deviation in meters until failsafe or crash											
	x in meters				y in meters				z in meters			
	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std
Stale	34.46	0.54	15.43	15.97	17.24	0.03	6.73	7.60	2.62	0.09	1.16	1.19
Default	4.51	3.89	4.30	0.23	39.65	39.65	39.65	0.00	4.88	4.85	4.86	0.01
Erroneous	98.83	0.44	30.39	29.29	125.12	0.68	26.61	36.34	54.05	8.43	32.34	18.76
Absent	58.26	57.24	57.76	0.37	138.14	138.11	138.13	0.01	53.12	53.07	53.10	0.02

detect, we notice that the failsafe requires an average of 5 seconds to detect the anomaly and trigger the land mode which is a large amount of time (recalling that the minimum time to detect is 1 second). The deviation results show that the attack deviates the drone from its trajectory, 1 to 15 meters on average. The standard deviation shows that the attack-induced behavior depends on the value used for the stale data. Each emulation run is not fully deterministic as there is simulated noise, for this reason, each run will start the stale data with a different value, which will cause different behavior on the controller.

Default Value Data. In this case, the SDA-induced behavior triggers the failsafe, but this time the drone crashes in every emulation as the controller cannot stabilize the vehicle after the attack. The induced behavior is consistent among each repeated experiment as we can observe a low standard deviation. This is explained by the fact that the transmitted data is always the same between runs. The time to detect is quick, as the failsafe is triggered consistently in 1.22 seconds (vs. 1 second minimum detection time).

Erroneous Data. In this case, the received data at the controller follows a normal distribution. This causes different behaviors in the drone as each sample received during the attack is random. The attack triggers the failsafe in the 27% of the cases (meaning that the attack is undetected most of the time). The times that the anomaly is detected, the drone crashes. When the attack is not detected, the drone crashes in the 45% of the cases and recovers the 27% of the times. When the anomaly is detected the detection requires on average 1.76 seconds. The impact on the controller causes deviations that

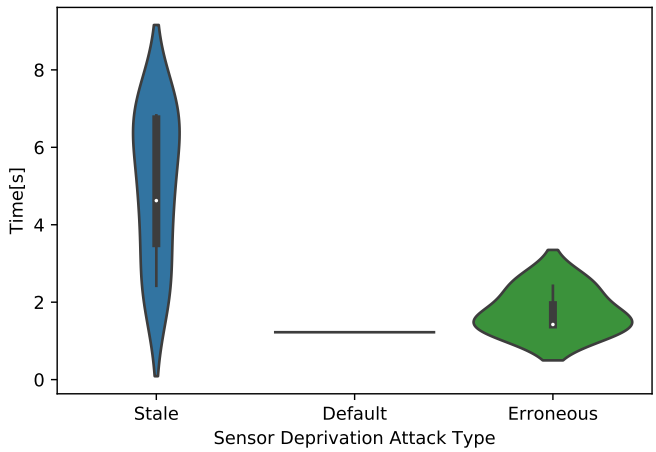


Figure 4.7: Time to Detect: violin plot representing the distribution.

span from 26 to 32 meters on average.

Absent Data. In this case, the controller stops receiving sensor updates. In our emulations, the drone crashes 100% of the time without triggering the failsafe. The induced behavior is consistent among the runs (i.e., a low standard deviation of the induced maximum deviation).

4.5.4 Summary

In relation to **RQ4.3**, we formalized the SDAs and hypothesized four possible sensor behaviors resulting from the attack. In relation to **RQ4.4**, the emulation results of SDAs show the severe impact of such manipulation on the physical process, which causes drone deviation from the expected path and crash. Moreover, we characterized the behavior of the failsafe mechanism in reaction to the SDAs. Our results show that Stale data attacks require a longer time to detect and the drone can safely land after detection, Default data attacks are quickly detected but the drone will crash, Erroneous data are most of the time undetected and the induced behavior varies (although most of times the drone crashes) and Absent data are always undetected and cause drone crash. Given the promising results of such attack vectors, we are now motivated to explore how such manipulations can be launched on real COTS hardware and which of the four hypothesized sensor behaviors occur in practice.

4.6 Realizing SDA in Hardware

After formalizing and introducing the Sensor Deprivation Attacks, we explore the feasibility of the attacks on real-world flight controllers. We address the following questions, **RQ4.5**: How can the Sensor Deprivation Attacks be practically launched on real modern flight controllers? **RQ4.6**: Which practical behaviors are observed and

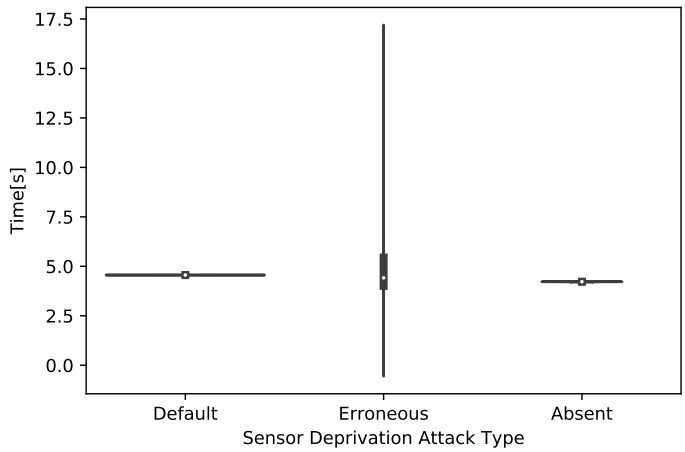


Figure 4.8: Time to Crash: violin plot representing the distribution.

what is their relation to the four hypothesized SDAs? **RQ4.7**: Are the induced SDA detected by State-of-the-Art detectors for UAVs?

To answer the outlined questions, we propose two attack strategies that allow achieving SDAs. We test these two strategies on four COTS flight controllers running the Ardupilot firmware, targeting four different IMU sensors. Moreover, we test our attacks on the Crazyflie drones equipped with the control firmware `2023.02` released in February 2023. Finally, we evaluate if the proposed attack strategies are detected by prior work anomaly detectors.

4.6.1 SDA via Sensor Reconfiguration

We propose to exploit two device re-configuration modes to achieve the SDAs on COTS flight controllers.

Sensor Suspend. The first methodology that we propose is the sensor suspend attack. Sensors allow suspension (also referred to as power-save mode, depending on the manufacturer) for power-saving reasons. In this mode, the sensor stops updating the sensor reading register. By doing so, the control software will continue its normal execution but the information received from the external sensor will be no more reliable (outdated, wrong, or absent depending on the manufacturer's specified behavior and the controller's behavior).

Sensor Sampling Frequency. The second methodology that we propose is the sensor sampling frequency attack. Since the sensors commonly found on modern flight controllers are general-purpose (e.g., the same IMU chip can be used in various contexts, for example, in smartphones, activity trackers, etc.) they can be programmed to accommodate different purposes. For this reason, the sampling frequency of the sensor can be changed based on the task. In flight control software sensors are often configured to run at the highest sampling frequency, this allows precise control of the aircraft. An attacker can resort to this device configuration to decrease the update rate of the sensor,

Table 4.4: Summary of IMU behaviors observed during Suspend mode attack.

IMU Model	Behavior	Accelerometer Value X		Y		Z		Behavior	Gyroscope Value X		Y		Z	
		μ	σ	μ	σ	μ	σ		μ	σ	μ	σ	μ	σ
BMI055	Absent Data	-	-	-	-	-	-	Default	-0.0025	0.0	-0.002	0.0	0.0012	0.0
BMI270	Default	141.53	0.0	141.53	0.0	141.53	0.0	Default	31.48	0.0	31.49	0.0	31.49	0.0
ICM-42688-P	Absent Data	-	-	-	-	-	-	Absent Data	-	-	-	-	-	-
MPU6000	Default	0.2477	1.058	68.9905	14.563	-0.3031	1.295	Default	0.0254	0.0	-0.0346	0	0.1211	0

in this way the sensor readings are updated less frequently, and the controller will rely on outdated sensor readings or wait until the next sample is collected.

4.6.2 On-hardware Evaluation

To verify the behavior of SDAs via reconfiguration attacks over the control algorithm, we run experiments on multiple COTS flight controllers and evaluate how different IMUs behave when reconfigured to the suspend mode and frequency attacks. We also evaluate how the control software reacts to those reconfigurations. To test the IMU SDAs, we flash the flight controller with Ardupilot, enable the logging, place the flight controllers on a flat surface, and launch the IMU reconfiguration while the flight control software runs on the board. For analysis purposes, we instrument the flight controller code to log attack start and attack stop events and low-level sensor readings. We then analyze the collected log files to assess the impact of sensor reconfiguration.

The attacker induces a change in the IMU configuration by sending a command through the SPI bus, more specifically, the attacker will issue the re-configuration command to the target IMU when the attack starts and resets the IMU to the previous configuration once the attack stops. The flight controllers used in our evaluation are i) Kakute H7 Mini equipped with the BMI270 IMU; ii) Pixhawk 6C with a dual-IMU configuration, a BMI055 IMU, and an ICM-42688-P IMU; iii) Omnibus F4 with a MPU6000 IMU. All the considered IMU peripherals connect to the MCU via the SPI bus. For each IMU we consult the publicly available manufacturer's datasheet and identify the register/value pair required for device reconfiguration.

Moreover, to assess the impact of the attacks in the real world, we evaluate the effects of the attacks on the Crazyflie V2 quadcopter drone, equipped with BMI088 IMU and an optical flow sensor for stabilization.

4.6.2.1 Effects of IMU Suspend

BMI055. Figure 4.9 shows the results of the suspend reconfiguration on the BMI055 sensor. IMU is composed of two sensor modules, the accelerometer (first plot in the figure) and the gyroscope (second plot). The accelerometer reacts by stopping sending values to the MCU (Absent data), while the gyroscope reacts to the configuration change by reporting a constant value (Default Value data). Table 4.4, reports the default value that was sensed on each axis during the attack. The Ardupilot controller reacts to the (partially) Absent data by applying the hold strategy on the last observed value, for this reason, the attitude estimation continues despite no accelerometer samples being

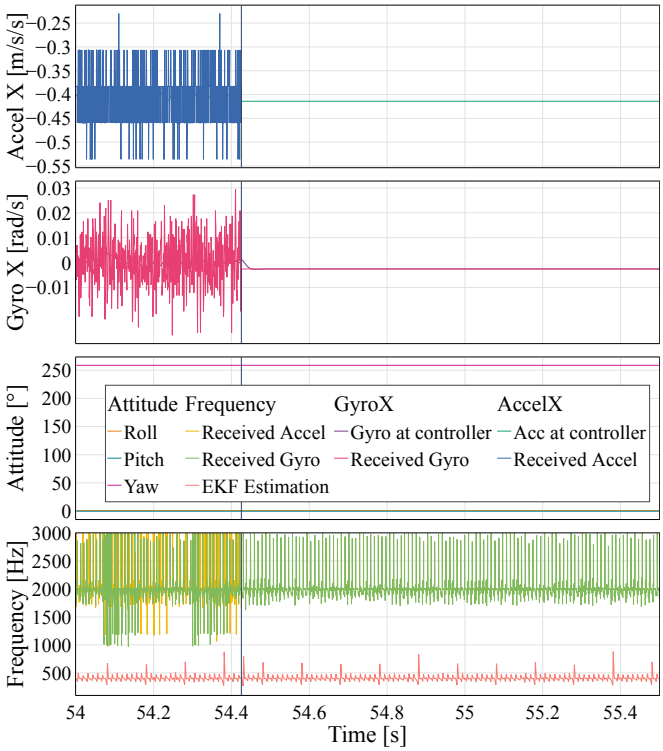


Figure 4.9: BMI055 - Suspend attack

received. As we can notice from the last plot in Figure 4.9, the frequency of the EKF estimation is not affected by the reconfiguration attack.

BMI270. Figure 4.10, shows the sensed behavior when starting the suspend attack on the BMI270. In this case, both the sensor modules react to the reconfiguration with the Default Value Data, the reported value for each module is an out-of-range value (see Table 4.4). Ardupilot accepts this received value as legitimate and this will induce an error in the attitude estimation. The attitude estimator will believe that the drone is flipping while the drone is steady on a table. Also in this case the reconfiguration does not affect the sensor and EKF frequency, leaving the controller scheduler unaltered.

ICM-42688-P. In this case, the IMU reacts to the suspend attack by stopping transmission of both the gyroscope and the accelerometer sensor updates. Since no data is received at the controller (for both sensors), no state estimation will be performed (see Figure 4.11). This result shows the resource dependency between the IMU and the state estimation. State estimation is part of the `fast_loop()` tasks, which are executed with the highest priority by the flight control scheduler with a frequency of 400Hz. Despite the control flow inside the flight controller remains unaltered, the unavailability of IMU data induced by the attack will make the highest priority task in Ardupilot being not executed.

MPU6000. As reported in Table 4.4 the MPU6000 behaves similarly to the BMI270 and will start reporting default values; those values are close to zero. In this case, after the reconfiguration, the sensor will also increase its responsiveness time, and as we can

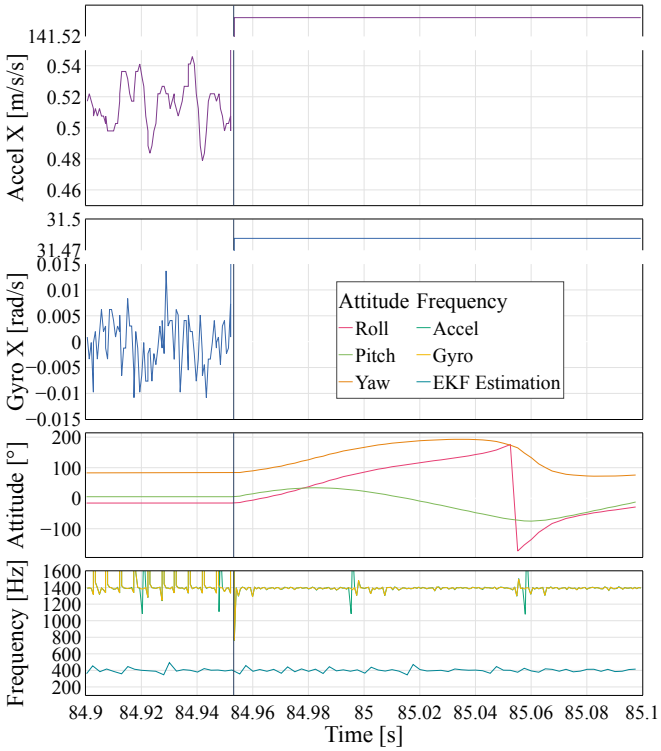


Figure 4.10: BMI270 - Suspend attack

notice in the last plot of Figure 4.12, the attitude and EKF frequency drops from 400Hz to 100Hz.

4.6.2.2 Effects of IMU Sampling Frequency

To test the frequency behavior, we set one by one all the possible allowed sampling frequencies in the target IMU.

The sensor is by default configured to operate at the highest sampling frequency (e.g., for the BMI 270, 3200Hz for the gyroscope and 1600Hz for the accelerometer). Then, we reconfigure the sensor to operate at a lower frequency. Figure 4.13, shows the impact of the frequency change compared to the Kalman Filter estimation frequency of the system when performing experiments on the BMI270 sensor, Figure 4.14 shows the impact on BMI055, Figure 4.15 shows the impact on ICM42688, and Figure 4.16 shows the impact on MPU600. In Ardupilot, the Kalman Filter prediction is scheduled for computation at a frequency of 400Hz. As long as the sensor sampling frequency is higher than 400Hz, the attack does not show any consequence on the controller. When the sensor frequency is reduced below 400Hz, we can observe that the Kalman Filter estimation gets affected by the sensor frequency and gets slowed down. This has severe consequences on the scheduler and drone control. In a cascading effect, the drone attitude estimation will not produce fresh outputs and consequently, no control commands are computed. This effect can be seen as a realization of intermittent Absent data values or as a Stale Data attack where the duration of the attack is $\frac{1}{\text{sensor frequency}}$.

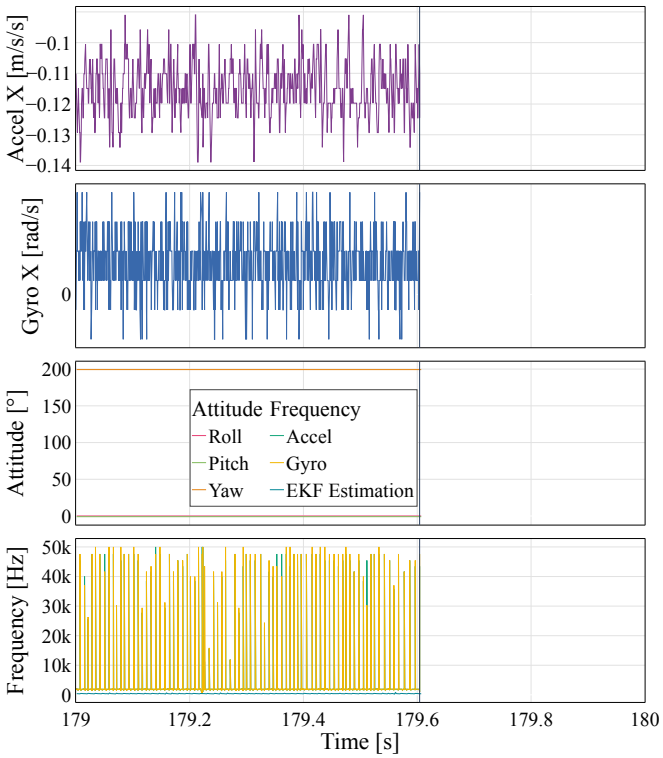


Figure 4.11: ICM42688 - Suspend attack

Moreover, we observed scheduled tasks in the ROTS being slowed down due to deadline misses (e.g., data transfer to the ground station will slow down and barometer sensors being sampled less frequently). In the next paragraph, we comment further on the observed consequences of our SDAs.

In our experiments, for all the considered IMUs, we were able to slow down the sampling frequency impacting the Kalman Filtering operations. Table 4.5, reports the lowest possible frequencies that can be set on each IMU.

Explanation for observed delay effects. Given the severe consequences of IMU sampling frequency on EKF frequency, and other data sampling (e.g., the barometer) we investigated the root cause of this behavior by looking into the ArduPilot codebase. We observed that the scheduler loop is continuously running inside an infinite while loop. This scheduler loop waits for new data from IMU and also performs vital tasks (81 in total) such as running the EKF state estimator or attitude controllers and checking if the UAV has landed or crashed. Differently from what we expected, although the reading of data from IMU is performed asynchronously, when the `sensor frequency < scheduler frequency` the scheduler loop waits for new IMU data to continue its execution. Since the IMU sampling frequency is influenced by the SDAs, all the vital tasks are not performed until a new reading is obtained from IMU. Our SDA exposes this design flaw, which makes it possible to delay the ROTS execution inside the MCU from an untrusted remote sensor.

Lowest frequency for control. Using the Ardupilot SITL simulator, we test the

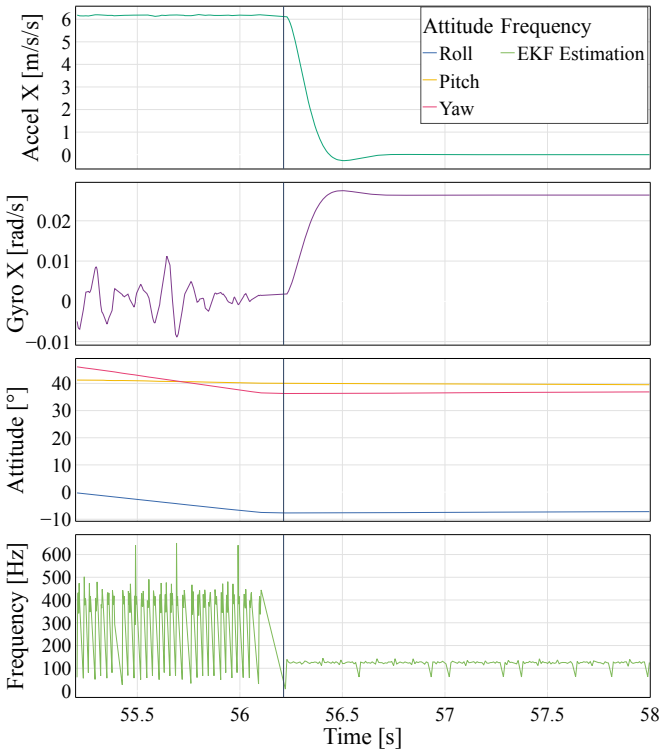


Figure 4.12: MPU6000 - Suspend attack

frequency change attack to verify the impact of the sensor frequency change on the flight and observe which is the frequency below which the flight controller will fail in controlling the drone. In our evaluation, we found that below 200Hz the drone becomes unstable, but it manages to continue the mission. Instead, below 150Hz the controller will fail to stabilize the drone and will crash on the ground. All the tested IMU allow frequencies lower than 150Hz.

Table 4.5: Summary of IMU behaviors observed during IMU frequency attack

	Lowest frequency (Hz)			
	Accelerometer		Gyroscope	
IMU Model	Tested	Allowed	Tested	Allowed
BMI055	15.56	7.81	32	32
BMI270	12.5	0.78125	25	25
ICM-42688-P	12.5	1.5625	12.5	12.5
MPU6000	100	50	100	50

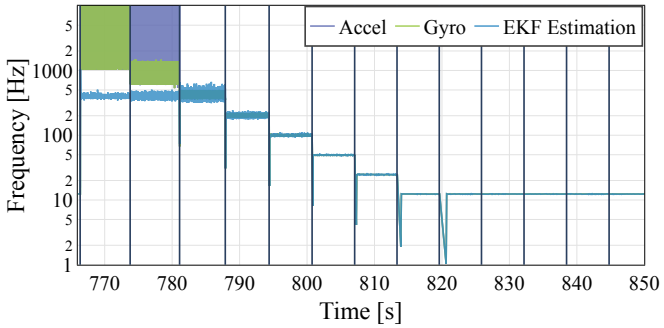


Figure 4.13: BMI270: Dependency between the IMU sampling frequency and the attitude estimation frequency. Below 400Hz, state estimation slows down to the frequency of the sensor.

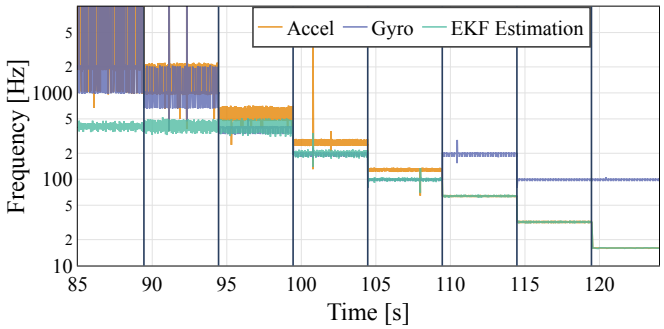


Figure 4.14: BMI055: Effects on main loop speed

4.6.2.3 End-to-end Attack Demonstration

To further evaluate the impact of the two attack strategies, we implement them on the CrazyflieV2 platform. A video demonstrating the attack impact can be accessed at this url `https://youtu.be/Egnw2dHKqw0`. The crazyflie device is hovering above the ground and it will continue doing so (baseline in the video) unless an attack is started. In BMI088, the suspend produces absent data behavior, while for the frequency attack, we configure the gyroscope to operate at 32 Hz, and the accelerometer to 12.5 Hz. As we show in the video, the impact of the suspended attack results in the drone crashing on the ground, while the frequency attack (which lasts 5 timesteps) causes the drone to deviate (with a longer duration of the attack, the result is similar to the suspend attack).

4.6.3 Detecting SDAs

We evaluate SDAs on flight controllers with the state-of-the-art security reference monitor M2MON [107] (see Section 4.2.4). In order to detect an attack, M2MON continuously monitors MMIO (Memory-mapped I/O) address range of the MCU with 3 different metrics:

1. Access list: List of MMIO address access

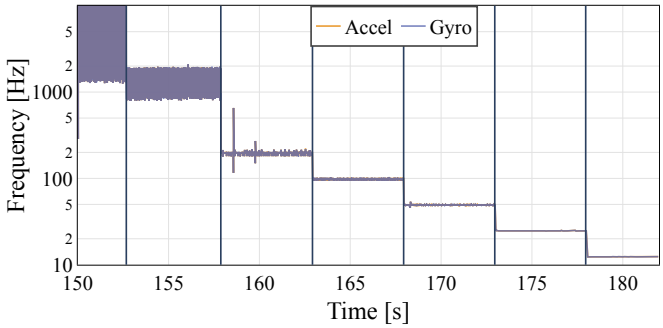


Figure 4.15: ICM42688: Effects on main loop speed

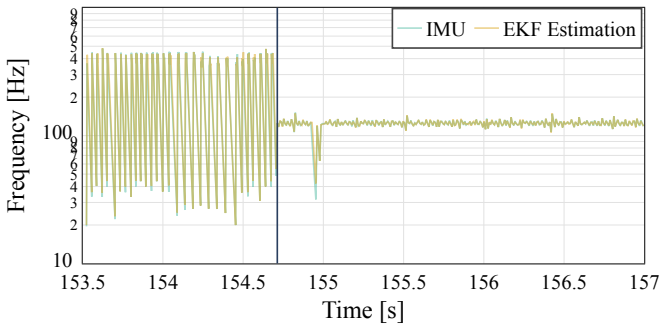


Figure 4.16: MPU6000: Effects on main loop speed

2. Access chain: Order of MMIO address access

3. Access frequency: Frequency of MMIO address access

Any deviation from regular access patterns indicates the presence of malicious actors. Additionally, in order to secure Extended Kalman Filter (EKF) from attacks on the real-time operating system (RTOS), M2MON runs EKF separately from the RTOS. Since full M2MON implementation is not available for the current ArduPilot project, first, we re-implement these monitors for ArduPilot. Then, we show that SDAs can remain undetected or delay detection during run-time.

4.6.3.1 Monitoring MMIO Access

In the case of Kakute H7 Mini (STM32H743), onboard BMI270 IMU is performing communication with SPI4 of the MCU. Upon referencing the datasheet of STM32H743 we find that transmit data register (TXDR) for SPI4 is located at address `0x40013420` and receive data register (RXDR) for SPI4 is located at address `0x40013430`. These registers are responsible for exchanging data with the IMU. Therefore, we inject counters into low-level SPI drivers for the ArduPilot firmware and count the number of accesses for the specified MMIO addresses responsible for handling SPI4.

4.6.3.2 Anomaly Detection

In an SDA the list and order of MMIO address access always remains the same as the flight controller firmware is not modified by the attack and it will continue querying the IMU regularly via SPI. Therefore, the attack remains undetected under the first two M2MON metrics. Further, we investigate the change in the frequency of MMIO address access. To achieve this, we first calculate the maximum and minimum MMIO access frequency for 300 seconds during the regular operation of IMU. To validate if MMIO access always remains within the calculated boundary, we compare it with a different 60 second trace of regular IMU operation. Finally, we perform sensor suspend and sensor sampling frequency attacks (at 12.5 Hz and 100Hz) individually on our IMU for 60 seconds and use the previously derived boundaries to detect SDAs. We observe that the regular operation of IMU suffers from transient changes in access frequency and this leads to false positives even when there is no attack happening. The detector has an accuracy of 0.999988 when monitoring idle MCU.

Sensor Suspend Attack Detection. In this case, the detector has low accuracy of 0.0002 when applied on suspend attacks. For this reason, we argue that this attack remains undetected. We attribute that sensor suspend attacks remain undetected by M2MON metrics since there is no change in communication patterns with the IMU. Moreover, monitoring the values that are set into the registers will not be sufficient to detect the device reconfiguration attack, the attacker issues the reconfiguration through an Out of Bound channel (e.g., CANFlict [60]) and not through the monitored MCU registers.

Sampling Frequency Attack Detection. In this case we observe that the detector has an accuracy of 0.9377 for 100 Hz and 0.9923 at 12.5 Hz re-configuration. This is because a sampling frequency attack leads to a decrease in communication frequency with the MCU. We argue that while the detector would be able to observe a change in access frequency, this detection will be delayed by the lower frequency of the IMU as there are 128 missing consequent observations when changing the IMU frequency from 1600Hz to 12.5Hz (we note that in the case of Absent Data, all the observations will be missing). Moreover, M2MON-EKF would wait for subsequent IMU readings for state estimation delaying the overall operations of M2MON (32 skipped state estimations, i.e., 12.5Hz vs 400Hz). As already noted by Jang et al. [102], induced higher time to detect the anomaly will consequently reduce the probability of attack remediation (failsafe/parachute) that leads the drone to crash. We would highlight that since the implementation of M2MON-EKF is the same EKF implementation in ArduPilot, moving EKF implementation outside RTOS offers no additional security guarantees against SDAs, as demonstrated in Section 4.5.3.1, Absent data remain undetected from the EKF based failsafe mechanism.

Process-based detection. Similarly, we argue that the time to detect increases also in the case of other proposed process-based anomaly detectors(e.g., Control Invariant [44] and SAVIOR [158]) which rely on sensor observation to detect anomalies. The delay induced by the proposed frequency attack will prevent the anomaly detection signal from reaching the detector, delaying or preventing the alarms (similarly to what is observed with absent data against the EKF failsafe, see Section 4.5.3.1).

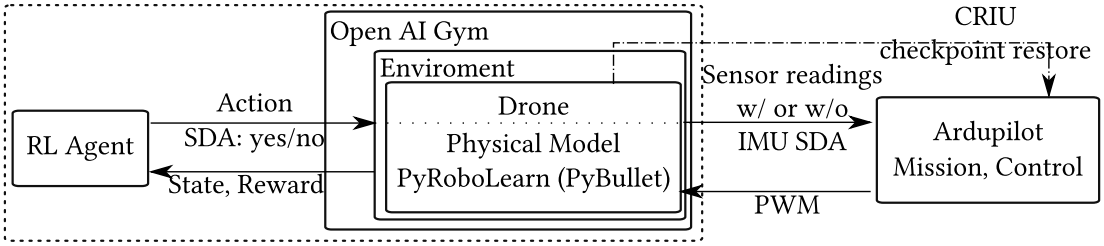


Figure 4.17: Controllable attack synthesis

4.6.4 Summary

To answer **RQ4.5**, we proposed the sensor suspension and the sensor sampling frequency attacks via sensor reconfiguration, to answer **RQ4.6** we performed hardware experiments to characterize the behavior of IMUs when targeted with the two proposed reconfiguration techniques and their implications on the control algorithm, our results show that the proposed techniques will result in three main behaviors (consistently with the behaviors hypothesized in Section 4.5, Absent, Stale and Erroneous data). Surprisingly we discovered that the sensor sampling frequency attack will force the attitude estimation to slow down to the frequency of the sensor and impact the whole RTOS scheduler. Finally, to answer **RQ4.7**, we evaluated the proposed attacks against prior work anomaly detectors and showed that the proposed suspend attack will go undetected while the frequency attack will slow down the state estimation and consequently delay the detection.

4.7 Deterministic Controllability

In the previous sections, we introduced the SDAs and evaluated their impact on drone flight, moreover, we showed how those attacks can be practically performed on COTS flight controllers. In this section, we investigate the following questions **RQ4.8** How UAVs can be adversarially controlled via SDAs? To answer this question, in this section, we investigate the controllability of the proposed SDAs and propose an attack synthesis methodology to adversarially drive the drone via SDAs.

4.7.1 Drone Deviation

When launching the SDA on the system, the last set of sensor readings indicates the attitude that the drone has before the attack starts. As we showed in Section 4.6, the SDA will induce the state estimation to be incorrect or delayed. This produces the wrong actuation to occur in the system. We are interested in understanding if this wrong actuation can be adversarially exploited to deviate the drone to an adversary's desired location.

When the SDA starts, the last computed attitude is the last correct estimate of the drone pose. The actuation command performed will reflect the correction w.r.t. the expected trajectory of the drone. In real-world control systems, the effect of measurement

and environmental noises will make the PID controllers continuously perform an error correction w.r.t. a target value.

We assume the case of a stale data SDA, the last observation before the attack is received by the controller during the attack. This will induce the controller to apply the same correction via actuators. The attacker monitors which action is being performed at the drone (correct to left, correct to right), and strategically starts a stale data attack to keep the current actuation being applied. This will result in the drone turning towards the attacker's desired direction.

4.7.2 Attack Synthesis

We propose a framework for SDA controllability based on Reinforcement Learning. The proposed framework allows the attacker agent to adversarially deviate a drone toward an attacker's desired location. To reach the desired location, the attacker decides whether the SDA should be performed each time.

4.7.2.1 Attack Synthesis Framework

Figure 4.17, summarizes the framework architecture for our evaluation. We create a custom Open AI Gym [25] environment to allow the interaction between the Reinforcement Framework agent and Ardupilot. Ardupilot SITL allows for an external physical emulation [11] via UDP socket, we rely on this feature to enable the interaction between the reinforcement learning environment and Ardupilot. To initialize the Ardupilot controller in the environment, we use Checkpoint/Restore In Userspace (CRIU) for restoring the Ardupilot process. The Ardupilot image was taken while the drone was flying a mission, and following a straight line.

Learning is based on episodes (from the beginning to the end of an emulation). Each episode is composed of steps. In each step, the agent computes the SDA decision based on the last received state and transmits it to the environment. The physical model will receive the PWM input from Ardupilot, simulate the physical response, send to Ardupilot the updated sensor reading or the attacked sensor readings (based on the attack decision from the agent), and send the updated physical state to the agent, which starts the next step of the framework. The attack synthesis relies upon the following elements:

- *Goal.* The goal is an X, Y, and Z coordinate to which we aim to deviate the drone. (The goal is randomized at the beginning of each episode to guarantee the generalizability of the learned policy).

- *Action Space.* The action space is a boolean variable, which reports whether the IMU readings should be under SDA or not.

- *State Observations.* The observation space is 25 dimensional and continuous, with X, Y, and Z drone position in meters, quaternion orientation (4 values), Roll, pitch, and yaw angles in radians, the velocity vector in m/s (3 values), Angular velocity in radians/second(3 values), Motors' speeds in RPMs (4 values), the roll, pitch and yaw first derivative and X, Y, Z goal position of the goal.

Algorithm 1: Reward function

```
flips = 0
function Reward(state)
    reward = 0
    goalDist = √(DronePos − GoalPos)²
    ΔDist = prevDist - goalDist
    if droneUP() and closeToGoal(ΔDist) then
        reward = reward+1.5
    else
        reward = reward-0.5
    if !droneUP() and closeToGoal(ΔDist) then
        reward = reward+0.5
    else
        reward = reward-1.5
    if droneFilps() then
        flips = flips + 1
        reward = -flips
    else
        flips = 0
    if closeToPlannedPath() then
        reward = reward - 2
    if failsafeTriggered() then
        reward = -40
    if goalDist < 1 then
        reward = 100
    return reward
```

- *Reward Function.* We design a reward function to train the RL agent for our drone deviation task (see Algorithm 1). The reward function evaluates the Euclidean distance to the goal (`goalDist`) and the change of the drone distance to the goal (`ΔDist`). Moreover, it rewards the drone flying properly (*droneUP()*), and penalizes it if upside down or flipping. Finally, it penalizes the drone following the controller's desired path (i.e., not flying away from the planned trajectory), penalizes it if the Ardupilot failsafe is triggered, and awards it when the drone gets closer to the goal.

4.7.2.2 Experimental Results

We train the attack synthesis agent for 1M iterations (19 hours) using the TRPO algorithm [172], Figure 4.18 reports the mean reward over the evaluation set (tested every 10k iterations). As we can see the model over time increases the average reward function value, which means that the agent policy learned how to deviate the drone via SDAs.

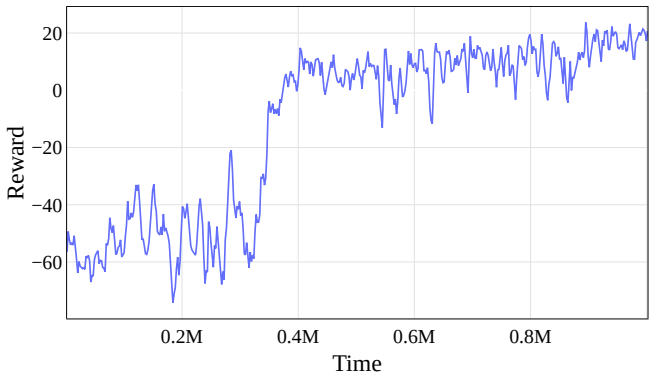


Figure 4.18: Attack synthesis training reward, TRPO algorithm, 1M iterations.

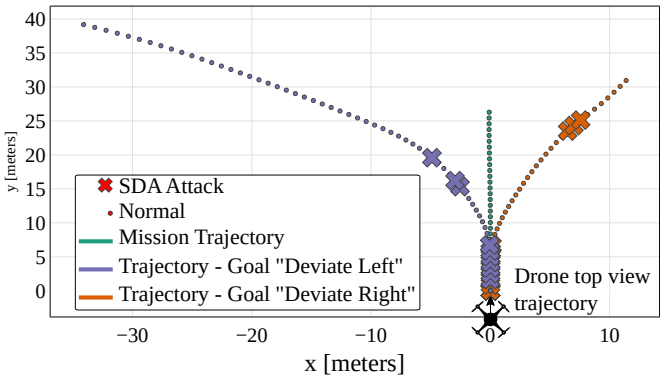


Figure 4.19: Sensor Deprivation Attacks attacks synthesis policy. By using the proposed Reinforcement Learning attack synthesis methodology, the drone is forced to deviate to the left or the right by the intermittent Sensor Deprivation Attacks.

We evaluate the trained agent's capability to perform the learned task. Figure 4.19 reports sample traces for the learned policy for the deviating left/right task, compared to the Ardupilot planned trajectory. The drone is flying on a straight trajectory and would succeed in following the trajectory unless SDAs are performed. The figures show the steps where the synthesis framework sent SDA action to achieve the desired goal. The results show that the proposed attack synthesis methodology can be used to strategically launch SDAs to adversarially fly the drone away from the planned trajectory, towards an attacker's desired location.

4.7.3 Summary

To answer **RQ4.8**, we proposed an attack synthesis methodology based on reinforcement learning. Our evaluation results show that the target UAV can deviate from the planned trajectory to the attacker's desired location. Compared to prior work attacks, the proposed framework represents an advancement in the attacker capability as previously demonstrated attacks will imply drone crashing [180, 102].

4.8 Discussion—Defending Against SDAs

The proposed attacks work by changing the configuration of the sensor chip, as we explained in Section 4.6, sensor configuration (and reconfiguration) occur via an unauthenticated bus. To the best of our knowledge, the hardware designs of sensor chips do not provide secure configuration options. Mitigate the proposed attacks at the sensor chip level would require:

- At a communication level, to enable the chips' secure configuration, there will be a need for an authentication layer on the serial protocol. This is challenging as it would induce computational and power consumption overheads on resource-constrained devices.
- At a hardware design level, remove configuration functionalities from the sensors. This way, the device configuration can not be overwritten by an attacker, at the cost of losing the freedom of configuration that enables general-purpose deployment of sensors. Or allow configuration only once after sensor boot and flush any upcoming configuration request. If the SDA occurs, either it occurs at boot (impeding the drone from arming), but it would not be possible to start it while the drone is flying.

Instead, at a software level from the controller side, possible mitigations would not take away the root issue of sensor reconfiguration but are easier to deploy in already deployed hardware. Despite the intrinsic limitations, attesting periodically that the sensor is correctly configured might help to mitigate the attack. Another possibility would require the ability to monitor the bus for anomalous requests. On the one hand, those solutions will introduce traffic overhead. On the other, software solutions open the discussion on how to remediate and recover once an attack is detected.

Attack Recovery. Once a reconfiguration is detected, reissuing the correct configuration commands would not be sufficient, as the attacker could reissue the reconfiguration command. Khan et al. [107] proposed to open the parachute as soon as an attack is detected. Choi et al. [43] proposed using software sensing techniques to recover from attacks in RVs. We argue that the recovery strategy depends on the context where the RV is deployed. Parachute opening might be seen as an extreme but conservative solution, while software sensing might allow drone recovery; if unsuccessful, it will lead to opening the parachute or crash.

4.9 Related Work: Control in Presence of Missing Deadlines

We discussed relevant related work about UAV security in Section 4.2. In this section, we report relevant related works from the systems engineering and control community.

In control theory, the problem of control in the case of missing observations or missing control commands has been theoretically formalized and studied. Sinopoli et al. [178] lay the foundations for this area of research, studying the convergence properties of Kalman Filter-based estimation in case of network faults. Their results show an upper limit of missing observations above which the state covariance matrix of the

filter becomes unbounded, making state estimation unreliable. Schenato et al. [170, 169] studied the problem of controllability in the case of lossy network communication and the performance of zero-input and the hold-input scheme control strategies. More recently, Maggio et al. [125] studied the stability guarantee properties of real-time control systems under sporadic deadline misses. Those related works study faulty behaviors of the control network with a stochastic approach, while we assume that the missing observations are attacker-intentional. Despite this difference, the theoretical boundaries introduced transfer to our attacker model.

4.9.1 Adversarial Reinforcement Learning

Recently, Reinforcement Learning techniques were applied to security-related contexts in attack and defenses. On the defensive side, Landen et al. [118] recently proposed a deep reinforcement learning agent to operate the power grid and perform anomaly detection, while Maiti et al. [126], proposes a reinforcement learning agent to perform vulnerability assessment of the power grid. On the attack side, Wu et al. [206] and Gong et al. [79] proposed policies to train deep reinforcement learning adversarial agents to attack state-of-the-art reinforcement learning agents.

To the best of our knowledge, we are the first to investigate the use of reinforcement learning to take over a control algorithm and perform adversarial control.

4.10 Conclusion on Sensor Deprivation Attacks

In this chapter, we presented novel Sensor Deprivation Attacks, exploiting sensor reconfiguration to achieve process destabilization. The proposed attacks differ from prior work attacks on UAVs (e.g., GPS spoofing) as the attacker does not need to craft attack packets to achieve the goal continuously. We first demonstrate malicious message injection on a serial bus testbed and successfully reconfigure victim sensors. To do so, we identified collision challenges and proposed a solution to achieve the message injection (addressing **RQ4.1** and **RQ4.2**)). Then, we formalize the hypothetical behaviors caused by SDAs (addressing **RQ4.3**), investigating their effects on real flight control software in emulation, and demonstrating that the attacks cause UAVs to deviate (up to 138 meters) from the expected trajectory and leading to crashes without triggering fail-safe mechanisms (**RQ4.4**). We then propose two techniques to launch SDAs on physical COTS flight controllers, namely suspend attack and sampling frequency attack (**RQ4.5**). To answer **RQ4.6**, we assess the impact of the proposed techniques on five flight control platforms and verify the attack-induced behavior on the sensor communication and flight control. We show that the attacks severely delay the drone attitude estimation (EKF estimation frequency depends on IMU frequency). We verify our attacks against prior work anomaly detection systems and show that the proposed attack techniques can evade completely (accuracy drops from 0.99 to 0.0002) or delay attack detection (**RQ4.7**). Finally, to address **RQ4.8**, we propose and evaluate an attack synthesis methodology to systematically deviate the drone toward an attacker's desired location via intermittent SDA. Our results show that an attacker can deterministically fly the drone to a desired location (for example, to deviate to the right or left).

Our findings demonstrate how the proposed SDAs threaten modern UAVs and what the consequences are on state-of-the-art control software. To defend against such attacks, we discussed possible countermeasures that range from modification to chip design to software-level mitigation.

Part II

Security Aspects of Process-based Anomaly Detection

5

Goals and Approach to Assess Anomaly Detection for CPS

Part I, investigated the current state of security in modern Cyber-physical systems highlighted the challenges in setting up secure CPS communication, and showed the effect on control stability caused by message manipulation. In Chapter 3, we showed that modern protocols that offer security features such as authentication and encryption present many challenges in the configuration, which often will result in insecure deployments. In Chapter 4, we demonstrated the impact on the control stability of a CPS when attackers manipulate the communication. Security challenges affect modern and legacy industrial protocols, and Industrial Control Systems are threatened by malicious actors, who aim to damage or destabilize physical processes [202]. Such attacks could be conducted through local physical-layer manipulation [117, 209, 193], compromise of local controllers [2, 71], or local network traffic [161]. Such activities produce anomalies in the physical sensor data that can be successfully leveraged for attack detection using Anomaly Detection systems. In this part of the thesis, we are interested in understanding the security of such anomaly detection systems, which are proposed to address those threats for CPS in a legacy-compliant way.

5.1 Introduction

A number of intrusion and anomaly detection schemes for ICS and in general for Cyber-Physical Systems (CPS) have been proposed in the literature to detect anomalies in CPS. Process-based anomaly detection [62] schemes leverage actuator and sensor data to detect anomalies in the process and operations of the system. We differentiate *model-free* and *model-based* anomaly detectors, depending on whether a physical model is leveraged by the scheme. Model-based detectors leverage the physical process directly (e.g., by using a set of linear or non-linear equations describing the process physics), those anomaly detection systems are harder to be constructed as the modeling of the process requires a plant-specific engineering effort [195, 44, 158]. Model-free detectors approximate the physical process indirectly and use Machine Learning [7], Deep Learning [77, 188, 112], System Identification [85, 9, 195], Support Vector Machines [41] and control invariants [5, 62] techniques for the training of the classifier/predictor.

Little is known about the resilience of those anomaly detection techniques against targeted manipulation, especially regarding classifier evasion [22]. If an attacker evades the anomaly detection system to conceal the true state and avoid or delay detection, can cause severe hardware damage or harm human beings. Evasion attacks in the context of ICS (which we call *Concealment Attacks* to distinguish them from the general case) face novel and specific challenges, which make standard Adversarial Machine Learning techniques [97] not directly applicable. Specifically, attackers are constrained in the number of features that they can manipulate (rather than the magnitude of the perturbation), attackers are bounded by the real-time constraints of the cyber-physical system, and adversarial perturbations must meet the physical properties of the process.

In this part of the thesis, we close the research gap about the resilience of anomaly detectors for CPS by proposing and evaluating concealment attacks. The content of this chapter provides the relevant information that constitutes the common ground of the upcoming Chapter 6, Chapter7, and Chapter 8. In Section 5.2, we start by reporting background information on the datasets used for evaluation, the CUSUM algorithm,

and classifier evasion. In Section 5.3 we summarize the state-of-the-art of anomaly detection for CPS, by providing an overview, taxonomy, and systematization of anomaly detectors proposed by prior work. In Section 5.4, we formally introduce and define Concealment Attacks, which is the approach that we propose to assess the security of anomaly detectors.

5.2 Background

In this Section, we summarize the relevant background information about ICS datasets, change detection algorithms, and evasion attacks relevant to this part of the thesis.

5.2.1 ICS process datasets

Datasets to evaluate detection schemes are collections of sensor readings and actuator states collected from ICS testbed or process simulation [189, 101, 129, 76]. Data are usually organized into two data captures. The first *train dataset* is collected during normal operating conditions, and the second *test dataset* is collected while anomalies caused by an attacker occur. Each dataset is labeled with the ground truth that reports for each time step if the system is 'anomalous' or 'safe'. We now introduce three popular ICS process datasets that we use for training, testing, and evaluation of attacks against ICS detectors.

5.2.1.1 BATADAL Dataset

Batadal dataset was released as part of the Batadal competition [189], which ran between 2016 and 2017. The dataset is generated through epanetCPA [186], a Matlab toolbox for simulating the hydraulic reaction of a water distribution system under attack. The BATADAL competition was based on three datasets: the first contains data collected by simulating 365 days of normal operations, and the second and third represent a collection 14 attacks (7 attacks per dataset). The details of the attacks can be found in [189]. These datasets contain readings from 43 sensors: 7 tank water levels, 12 inlet and outlet pressure for one actuated valve, and all pumping stations, and 24 flow, and status variables. The variables are continuous except for the categorical status variables.

The original attack dataset contained sensor data readings that were manually concealed. For that reason, we could not use the original attack dataset directly (as we wanted to add concealment ourselves). Instead, we re-created the attacks (and resulting sensor data) from the BATADAL dataset for this thesis using the original setup, without any manual concealment. In our new version, the data are collected from sensors every 15 minutes.

5.2.1.2 WADI Dataset

The WADI dataset was collected from the Water Distribution (WADI) testbed at the Singapore University of Technology and Design [8]. The WADI testbed is composed of three cascading processes: P1 is the primary supply and analysis stage, P2 is the elevated reservoir with domestic grid and leak detection stage, and P3 is the return

process stage. Three 3 PLCs control the WADI testbed, monitoring 103 sensors in the network. Moreover, WADI is equipped with a SCADA system. For anomaly detection, we use the 82 sensors data collected at P1 and P2, since P3 is implemented for recycling water. The train dataset contains data from 14 days of normal operations. The test dataset contains 15 attacks on physical processes over 48 hours. This dataset is available on request [101].

5.2.1.3 SWaT Dataset

SWaT [129] is a water treatment testbed located at the Singapore University of Technology and Design. It consists of a six-stage process for water treatment. Those six stages are controlled by interconnected PLCs, connected to Human Machine Interfaces (HMIs), a Supervisory Control and Data Acquisition (SCADA) workstation, and a Historian. The SWaT dataset is a collection of 51 sensors and actuators from 11 days of operations; 7 days were collected during the system's normal operation and 4 days were collected while 41 attacks were launched on the system.

5.2.2 CUSUM

The Cumulative Sum (CUSUM) is a nonparametric statistic used for change detection. Given a set of sensor readings coming from the physical process, in the anomaly detection problem, we are interested in detecting a change in the process generating the sensor readings and triggering an alarm as soon as possible [33].

Given the values of a time series $Z(k) = z(1),\ldots,z(n)$, the goal is to determine whether the values come from the normal system operations (hypothesis H_0), or are anomalous (hypothesis H_1).

The algorithm is designed to minimize the number of samples (N) required to detect a change in the process (i.e., detect that the sensor readings that were generated under the H_0 hypothesis are now generated under the H_1 hypothesis).

The CUSUM is defined as:

$$S_{k+1} = \begin{cases} 0 & \text{if} \quad t = 0 \\ (\log \frac{p_1(z(k))}{p_0(z(k))} + S(k))^+ & \text{if} \quad t > 0 \end{cases} \tag{5.1}$$

where $(a)^+ = a$ if $a \geq 0$ and 0 otherwise. An alarm is raised at time N when $S(n) \geq \tau$ (where τ is a threshold).

Cardenas et al. [33], proposed to use the non-parametric statistics to compute the CUSUM for the anomaly detection case (i.e., p_0 and p_1 are non-parametric). Instead, they assume $\mathbb{E}[Z(k) < 0]$ for H_0 hypothesis and $\mathbb{E}[Z(k) > 0]$ for the H_1 hypothesis. Specifically, Cardenas proposes to use the residuals of the process $r = |y(k) - \hat{y}(k)|$ (i.e., the difference between the predicted sensor readings at the actual sensor readings at time k), and define $z(k) = |y(k) - \hat{y}(k)| - b$ where b is chosen to satisfy $\mathbb{E}[|y(k) - \hat{y}(k)| - b < 0]$.

5.2.3 Evasion Attacks

In Adversarial Machine Learning, an evasion attack refers to the use of *adversarial examples* to control the output of a machine learning model. Adversarial examples are

generated by an attacker based on the knowledge they have of the target model.

Evasion attacks and defense were proposed in various machine learning applications such as image classification [185], speech recognition [36] and malware detection [84].

Biggio et al. [23] characterized adversarial machine learning attacks using a tuple representation of the attackers' knowledge of the system under attack, the knowledge of the training dataset $\mathcal{D}$, the knowledge of the feature set $\mathcal{X}$, the knowledge of the learning algorithm f, and the knowledge of trained parameters w. An Adversarial Machine Learning attacker has complete or partial knowledge of the elements in the tuple; partial knowledge of an element is denoted with $\hat{\mathcal{D}}$, $\hat{\mathcal{X}}$, $\hat{f}$ and $\hat{w}$. Using the tuple, three attack scenarios are characterized by the authors: Perfect-knowledge white box attackers $(\mathcal{D}, \mathcal{X}, f, w)$; Limited-knowledge gray box attacks $(\hat{\mathcal{D}}, \mathcal{X}, f, \hat{w})$; Zero-knowledge black box attacks $(\hat{\mathcal{D}}, \hat{\mathcal{X}}, \hat{f}, \hat{w})$. Attacks are achieved by solving an optimization problem that minimizes the distance between the sample and the adversarial example e.g. by minimizing norms: L_0, L_2, L_∞. We use this notation to characterize our proposed concealment attacks.

5.2.4 Evaluation Metrics

Anomaly detection is a binary classification problem, and the goal is to classify the samples as 'anomalous' or 'safe'. To evaluate the performance of the anomaly detectors under attack, we observe how Accuracy Equation 5.2, Precision Equation 5.3, Recall Equation 5.4, F1-Score Equation 5.5 and False Positive Rate Equation 5.6 scores change before and after the concealment attack is applied to the data.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{5.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.4}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{5.6}$$

where TP stands for True Positive, TN for True Negative, FN for False Negative and FP for False Positive. In the anomaly detection task, we consider the 'safe' class the negative class, and the 'anomalous' class the positive class. Given the ground truth associated with the samples used for training and testing the model, the True Positives are the correctly classified positive samples (i.e., for the anomaly detection task the correctly classified 'anomalous' samples), the True Negatives are the correctly classified negative samples (i.e., the correctly classified 'safe' samples), the False Positives are the incorrectly classified negative samples (i.e., the 'safe' samples that are being classified as 'anomalous'), and the False Negatives are the incorrectly classified positive samples (i.e., the 'anomalous' samples that are being classified as 'safe').

Table 5.1: Taxonomy of the considered process-based anomaly detection. ○simulated process/data, ●real process/data, ◐simulated and real process/data.

	AR [85, 195, 9]	PASAD [9]	SFIG [62]	AE [77, 188, 112]	LTI [195, 158]	SVM [41]
Evaluation						
SWaT	●	●	●	●	●	○
WADI	-	-	●	●	-	-
Batadal	-	-	-	○	-	-
Private/Other	◐	●	-	-	●	-
Code Public	✗*	✓	✓†	✓	✗*	∼±
Analyzed in Chapter	7, 8	7, 8	7, 8	6, 7	8	8

*We re-implemented the methods and released the code publicly together with paper publication
† The code was initially not available, hence we re-implemented the method. After our re-implementation and evaluation were completed the authors released the code of the original implementation.
± Available upon request.

5.3 Taxonomy of Process-Based Anomaly Detectors

In this section, we provide a taxonomy of prior work on ICS anomaly detectors (see Table 5.1). For each considered detector we present a summary of how it works, on which dataset it was evaluated, and whether the detector code is publicly available.

In this thesis, we evaluate the security aspects of those target anomaly detectors, in Table 5.1 we report a mapping between the detectors and the chapter of the thesis where its security evaluation is presented.

5.3.1 Reconstruction-based Attack Detector

With the term Reconstruction-based Attack Detector, we refer to anomaly detection systems proposed in prior works [77, 112, 188], which share the same underlying idea of employing autoencoders architectures for anomaly detection.

Autoencoders are a class of deep learning architectures that were successfully applied in anomaly detection tasks for industrial control systems. In particular, fully connected (FC) [188], convolutional neural network (CNN) [112], and long short-term memories (LSTM) [77] autoencoder architectures were proposed to detect anomalies in process data. The general idea of those anomaly detectors is to train the Autoencoder model to reproduce sensor readings occurring in the system during normal operations (i.e., without attacks) by minimizing the Mean Squared Error (MSE) between the input and output layer of the network. Then a threshold is set over the MSE observed during the system validation. At test time anomalies in the sensor data presented at the input layer will not be reproduced in the output layer, this will produce a reconstruction error higher than the threshold and will trigger an alarm. CNN and LSTM architectures are trained taking into account the temporal evolution of the signal while the FC is not.

We formally describe the functioning of a Reconstruction-based Attack Detector.

The input to the autoencoder is $X = [\vec{s_{t-m}}, \ldots, \vec{s_t}]$, representing $m+1$ time-steps of $\vec{s} = [r_1, r_2, \ldots, r_n]$, which is an n-dimensional vector of sensor readings. The autoencoder trains the function (g) parameters (ϕ) that minimize following the Mean-Squared Error optimization problem:

$$\min_{\phi} \mathbb{E}_{(X,\vec{s_t})\sim\mathcal{D}} \left[\|g(\phi, X), \vec{s_t}\|_2^2 \right] \tag{5.7}$$

where the function g outputs an n-dimensional vector $\vec{o} = [v_1, v_2, ..., v_n]$, and v_i $s.t.$ $i \in \{1, ..., n\}$ represents the reconstructed value w.r.t. the input reading r_i^t. To decide if the system is under attack, the mean squared reconstruction error between observed and predicted features is computed ($\vec{e} = \vec{s_t} - \vec{o_t} = [d_1, \ldots, d_n]$). If the mean squared reconstruction (Equation 5.8) error exceeds a threshold θ, the system is classified as under attack (Equation 5.9).

$$\varepsilon(\vec{e}) = \frac{1}{n} \sum_{i=1}^{n} {d_i}^2 \tag{5.8}$$

$$f(X) = \begin{cases} \text{'anomalous' if } \varepsilon(\vec{e}) > \theta \\ \text{'safe' otherwise} \end{cases} \tag{5.9}$$

5.3.2 Autoregressive Model (AR)

Autoregressive models, Equation 5.10, are a representation used to deal with time-series data. Several prior works [85, 195, 9] considered the AR predictors to perform anomaly detection (either as a contribution or as a baseline for evaluation). AR models are a popular method used to model time series processes using linear equations starting from process data. The Autoregressive model AR(p) of a time series correlates the time series value (X) at time t with previous $t-p$ time series values. Specifically, an Auto Regressive model (Equation 5.10), tries to minimize the prediction error of sample X_t given the previous values ($X_0 \ldots X_{t-1}$).

$$X_t = c + \sum_{i=1}^{p} \gamma_i X_{t-i} + \epsilon_t \tag{5.10}$$

Where c is a constant, $\gamma_i, \ldots, \gamma_p$ indicates the parameters of the model and ϵ_t is white noise. The parameters $\gamma_1, \ldots, \gamma_p$ can be identified with several methods (e.g., ordinary least squares, or Yule-Walker equations [92]).

The general idea to use the AR model as an anomaly detector is to fit the Autoregressive model to time series data under normal operating conditions of the CPS, compute predictions, and analyze the residuals to identify anomalies in the sensor data. Different classification functions (detection statistics) were used to identify anomalies. The work by Urbina et al. [195] shows that stateful Cumulative Sum (CUSUM) (see Section 5.2.2) provides a better detection performance.

5.3.3 Linear Time Invariant Models

Linear Time invariant systems are a way to model input-output systems, which are characterized by linearity and time invariance. Linear Time Invariant models were

applied for anomaly detection in CPS [195, 44]. Equation 5.11, represents the State Space representation of an LTI system.

$$\begin{cases} s_{k+1} = As_k + Bq_k \\ x_k = Cs_k + Dq_k \end{cases} \tag{5.11}$$

Where $k := kT$ and T is the sampling time. $s_k \in \mathbb{R}^n$ is the state of the system, i.e., the variables (directly or indirectly observable) of the process. $q_k \in \mathbb{R}^p$ is the input to the system. $x_k \in \mathbb{R}^q$ is the output of the system. $A \in \mathbb{R}^{n \times n}$ is the state matrix, relates the state s_k and its update s_{k+1}. $B \in \mathbb{R}^{n \times p}$ is the input matrix, relates the system input q_k and the state update s_{k+1}. $C \in \mathbb{R}^{q \times n}$ is the output matrix, relates the state s_k and the measured output x_k. $D \in \mathbb{R}^{q \times p}$ is the feed-through matrix, relates q_k and x_k.

LTI model parameters can be derived either by knowing the system equations (model-based) or by applying system identification (using the n4sid algorithm [197]) on process data (data-driven model). Given an LTI model of a CPS, prior work proposed to apply the CUSUM algorithm performs anomaly detection. More specifically, the LTI is used as a one-step-ahead predictor, then the residuals between the predicted and observed values are classified by the CUSUM algorithm (see Section 5.2.2).

5.3.4 PASAD

The PASAD model proposed in the work by Aoudi *et al.* [9] is a model-free anomaly detector. The key motivation of this anomaly detector is to learn and obtain a mathematical representation of the regular dynamics/deterministic behavior of sensor signals in the ICS and spot any anomalous deviation. Singular Spectrum Analysis is used to analyze sensor readings and learn how the process behaves in normal conditions. Sensor readings are considered a univariate time series. Training is conducted on a set of *lagged* sensor readings, those readings are processed with Singular Value Decomposition, and projected into a Signal Subspace, where they form a cluster of points. At test time, test samples are processed with the same pipeline and compared against the centroid of the training data cluster in the Signal Subspace. If tested points exceed a distance threshold from the centroid, the system state is evaluated as anomalous. The MATLAB framework is available as open-source[1].

5.3.5 Systematic Framework for Invariants Generation (SFIG)

This work by Feng et al.[62] proposes a framework for the automatic extraction of process invariants, which are used to build a model-free anomaly detector. The framework applies data mining techniques to extract invariants from frequent itemset with multiple minimum supports. The framework consists of predicate generation and invariant mining. Predicate generation considers three kinds of predicates. Categorical predicates are generated according to actuators' states. Distribution-driven predicates, use Gaussian Mixture Models to identify the K probability distributions that describe the sensor value update (done per-sensor). Event-driven predicates are fitting Lasso regression models to capture the interplay between actuators' states and values seen by sensors.

[1] `https://github.com/mikeliturbe/pasad`

Predicates are combined to create sets of predicates that hold in a certain instant over the ICS, extracted via the CFP-growth++ [108] algorithm. Evaluation is conducted over WADI and SWaT datasets [101, 129]. After we re-implemented the detection method for evaluation (as it was not available), the detector was made available online on GitHub[2].

5.3.6 SVM

In the work by Chen et al. [41], the use of SVM is proposed to detect anomalies in CPS process data. The proposed SVM is trained on the water tank sensor readings (π,π') measured at d timesteps from each other to detect anomalous sensor reading temporal evolution. In the original formulation presented in the paper, the authors assume that anomalous data are available for training (differently from data split considered in the popular datasets 5.2.1).

To apply their proposed method on a dataset that contains exclusively benign samples in the training set, we switched to a one-class SVM classifier. ***Availability.*** This detector was made available to us by the authors of [41] upon request. We adapted it to work with the SWaT testbed dataset (originally it was proposed for the SWaT simulator).

5.4 Proposed Approach: Concealment Attacks

To assess anomaly detectors we propose concealment attacks. Concealment attacks will be used in the next three chapters to assess the resilience of the target detection mechanisms. This section presents the system model, and attacker's goal and capabilities used in a concealment attack, together with a formal definition of the attack.

5.4.1 System Model

We consider an Industrial Control System as depicted in Figure 2.2. The industrial architecture is organized according to the Purdue Enterprise Reference Architecture [205] (see Section 2.1.1.1). Level 0 consists of a number of sensors and actuators, connected to a controller in Level 1 (e.g., PLC). The controller forwards local control commands u and sensor readings y to Level 2, where the local area SCADA is deployed (the ICS consists of multiple different areas). At Level 3 the data gathered from the different Level 2 areas are aggregated and analyzed by a process-based anomaly detector (e.g., one of the detectors presented in Section 5.3) that uses the sensor data to classify the system state as anomalous or normal.

During an attack, the physical process will reach an anomalous state, which will be detected unless the attacker manages to conceal it (Figure 5.1). The anomalies themselves are out of the scope of this thesis as we use prior-work datasets containing anomalies (see Section 5.2.1)

[2] https://github.com/cfeng783/NDSS19_InvariantRuleAD

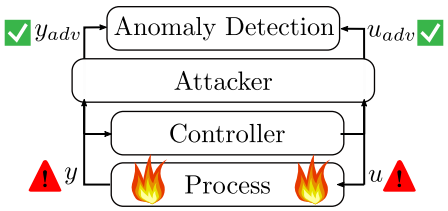


Figure 5.1: System and attacker model, we assume a physical process that is controlled by a controller and monitored by an anomaly detection system. The attacker wants to hide an ongoing anomaly on the CPS from the anomaly detection system.

5.4.2 Attacker capabilities

The attacker is assumed to have access to the ICS network, e.g., by physically attaching malicious devices to the network, intercepting communications to selected remote substations, or performing a Man-in-the-PLC [71] attack. The attacker can modify exchanged industrial traffic in transit, or compromise intermediate hosts (PLCs or SCADA) to change values being forwarded (y_{adv} and u_{adv}, see Figure 5.1). PLC communication with the SCADA system can be exploited to hide the real state of the system as practically demonstrated by Garcia et al. [71]. For example, in the Stuxnet attack [202] a compromised PLC was changing the rotation frequency of centrifuges of a nuclear process while reporting the correct frequency value to the anomaly detection to hide the anomaly.

5.4.2.1 Attacker Goal

The attacker's goal is to launch a concealment attack to hide an ongoing physical process anomaly from an anomaly detector. Sensor and actuator values from a CPS are logged and used for anomaly detection. Given an anomalous feature vector $x = (y, u)$ (i.e., sensors and actuators readings) collected at a certain instant in time, a binary classification function $f(x)$ that classifies system state as 'anomalous' or 'safe', the concealment attack looks for a feature perturbation δ that added to x produces target misclassification (Equation 5.12).

$$
\begin{aligned}
&\text{Given} && x = (y, u) \\
&\text{s.t.} && f(x) = \text{'anomalous'} \\
&\text{where} && y \in \mathbb{R}^n \\
& && u \in \mathbb{R}^m \\
& && x \in \mathbb{R}^{n+m}
\end{aligned}
\qquad
\begin{aligned}
&\text{Find} && x_{adv} = x + \delta \\
&\text{s.t.} && f(x_{adv}) = \text{'safe'} \\
&\text{where} && x_{adv} = (y_{adv}, u_{adv}) \\
& && y_{adv} = y + \delta_y, \delta_y \in \mathbb{R}^n \\
& && u_{adv} = u + \delta_u, \delta_u \in \mathbb{R}^m
\end{aligned}
\tag{5.12}
$$

In the next chapters, we will present three different frameworks to compute the perturbation (δ), based on different assumptions about the attacker's knowledge of the target anomaly detector, and evaluate its evasion performance on several tested anomaly detectors.

5.4.3 Attacker Space: knowledge of the target anomaly detector

Using the adversarial learning notation introduced in Section 5.2.3, a concealment attack is characterized by the attacker of the training dataset $\mathcal{D}$, feature set $\mathcal{X}$, learning algorithm f, and trained parameters w. In the ICS setting, the attacker can be characterized differently according to his knowledge of the attacked system. In order to explain our attacker model, we split the tuple $(\mathcal{D}, \mathcal{X}, f, w)$ into two: the Data tuple $(\mathcal{D}, \mathcal{X})$ and Defense tuple (f, w). We assume the attacker to be *unconstrained* or *constrained* w.r.t. the Data tuple, i.e., the sensor readings $\mathcal{X}$ that she can observe and manipulate and the data $\mathcal{D}$ that she eavesdrops. Moreover, we classify attackers as *white box*, *black box*, w.r.t. the Defense tuple, i.e., the knowledge of learning algorithm f, and trained parameters w.

Constraints over Data Tuple. we classify the attacker as:

- *Unconstrained* $(\mathcal{D}, \mathcal{X})$, the attacker can manipulate all n features in $\vec{x}$, and her perturbations are limited in terms of L_0 distance to be at most n.
- *Features Partially Constrained* $(\mathcal{D}, \hat{\mathcal{X}})$, we assume a constrained attacker capable of perturbing a subset k of n variables in $\vec{x}$, and her perturbations are limited in terms of L_0 distance to not exceed distance k.
- *Features Fully-Constrained* $(\mathcal{D}, \hat{\mathcal{X}})$, we assume a constrained attacker capable of observing and perturbing a subset k of n variables in $\vec{x}$, and her perturbations are limited in terms of L_0 distance to not exceed distance k.
- *Data Constrained* $(\hat{\mathcal{D}}, \mathcal{X})$, we assume a constrained attacker capable of eavesdropping on a limited quantity of process data that are used for training its attacks.

Knowledge of Defense Tuple. We classify the attacker according to their knowledge of the Defense tuple (f, w), as:

- *White box* (f, w), the attacker has a perfect knowledge of the target anomaly detection system and can access its variables, values, and thresholds (i.e. (f, w)). With the white-box knowledge of the system, the attacker can query the detection system and perform optimizations to compute the adversarial example. For example, the attacker can perform gradient signal-based optimization.
- *Black box* $(\cancel{f}, \cancel{w})$, the attacker does not have access to the anomaly detection system or its parameters. Differently from the black box attacker from Biggio et al. [23] $(\hat{f}, \hat{w})$, our attacker does not have the knowledge of f or its partial knowledge $\hat{f}$. We make this assumption because, in the CPS context, it would be challenging for the attacker to query the anomaly detection system (this is done for example in [192, 208, 40, 50]), as this would trigger an alarm on the anomaly detection system.

Given this taxonomy, the attacker can be classified for example, as *unconstrained white box*.

5.5 Related Work: Machine Learning Security

In this section, we discuss relevant work related to machine learning applications in CPS, their security, and adversarial machine learning in general. In this part of the thesis, we propose a taxonomy of anomaly detectors for cyber-physical systems and evaluate their resilience against the proposed concealment attacks.

5.5.1 Anomaly Detection in CPS

We summarize other related work proposed in the field of anomaly detection for cyber physical systems. Adepu et al [5] present a detection scheme based on the usage of process invariants. The study focuses on the so-called 'State-dependent' and 'State-agnostic' invariants. Experiments are conducted in the SWaT testbed. Ahmed et al. [7], tackle the problem of intrusion detection from the fingerprinting perspective. The main idea is to detect anomalies occurring in the system from changes in the noise in sensor readings. The paper proposes a mix of model-based and model-free techniques. To obtain a sensor fingerprint, Kalman filtering is used. The obtained prediction is compared with the real state to produce residuals. Those residuals are used to produce features to train a Support Vector Machine classifier. Implementation is not available as open-source. Hau et al.(2019) [90] consider the problem of false data injections in low-density wireless sensor networks. They develop a model-free anomaly detector that exploits temporal-attribute correlation among sensor readings. They extract pairwise feature cross-correlation and check if the stream of data statistically supports this correlation. When cross-correlation indices fail the test, an alarm is raised. Choi et al. [44] proposes a model-based detector based on PID controllers to approximate Robotic Vehicle's (RV) control logic. The PID controller predicts the system state, if the the prediction error surpasses a threshold an alarm is raised. The error is monitored in time windows of length w. w is tuned to overcome the latency that can be introduced by the PID. Quinonez et al. [158] use Extended Kalman Filters (EKF) to detect anomalies in Autonomous Vehicles in a model-based manner. Specifically, the system exploits the knowledge of the physical dynamics of the system to extract physical invariants. Those invariants are learned in an offline manner, collecting data from the vehicle in motion. The prediction of the EKF, combined with CUSUM change detection algorithm allows the detection of anomalies.

5.5.2 Attacks to CPS anomaly detection

Related work proposed the evaluation of anomaly detection systems against targeted manipulations. Feng et al. [61] propose the usage of Generative Adversarial Networks to produce stealthy manipulations for ICS detectors. The work by Zizzo et al. [214] evaluates adversarial examples in ICS by applying white-box attacks to LSTM detectors. Moreover, Kravchik et al. [111] proposed poisoning attacks against deep learning-based anomaly detectors for CPS, the proposed attacks poison the training data used by the detector. The injected malicious samples into the training will modify the resulting anomaly detection system behavior to hide anomalies. The work by Walita et al. [S2], proposes backdoor attacks against deep learning-based anomaly detectors for CPS. The

backdoor trigger is implanted through the actuator states in the training data. When the model is deployed the performance is not affected until the trigger is activated by the attacker. When the attack is triggered anomalous behavior in the system will go undetected.

5.5.3 Adversarial Machine Learning in CPS

Related work proposed adversarial machine learning attacks against machine learning pipelines used in CPS systems. Those attacks target the perception and localization pipelines used by vehicles to enable autonomy.

Attacks against Lidar-based perception. Cao et al. (2019) [30] proposed attacks against lidar-based perception, the proposed lidar spoofing attack injects points in the point cloud to create a fake obstacle in the scene. Sun et al.[184] proposed lidar-spoofing attacks and defenses, the proposed attack exploits geometrical transformations to inject fake obstacles in the scene, moreover they propose two techniques to detect such attacks based on looking at the density of the points in the point cloud, as the fake spoofed objects have a lower density then real obstacles. Hau et al.(2021a) [89], proposed Object Removal Attacks against lidar-based perception. The proposed technique injects points in the lidar point cloud to shift objects away from the victim and hence make obstacles disappear. Hau et al.(2021b) [88], proposed a mechanism to detect object removal attacks based on physical invariants (3D shadows). Moreover. Hau et al.(2022) [87], proposed the use of 3D shadow invariants to detect object hiding attacks. Cao et al.(2023) [28], proposed Physical Removal Attacks to hide objects in the lidar point cloud. Bhupathiraju et al. [21], proposed electromagnetic emission-based attacks against lidar perception used in autonomous driving, the proposed attack can cause object removal or obstacle injection in the scene.

Attacks against vision-based perception. Sato et al. [168], proposed attacks against vision-based line centering algorithms for autonomous vehicles. The authors propose the use of 'dirty road patches' to cause wrong perception of the street lanes and make vehicles take wrong turns and drive outside of the road. Lovisotto et al. [123] proposed projector-based attacks against traffic sign recognition for autonomous vehicles, the proposed methodology makes a deep learning-based traffic recognition system to misclassify traffic signs.

Attacks against sensor fusion pipelines. Cao et al.(2021) [29], proposed attacks against vision-lidar perception by generating obstacles that are concealed from both systems at the same time. Shen et al. [175], proposed attacks against the multi-sensor fusion used in autonomous vehicles to estimate the positioning. The proposed attack exploits transient low confidence in one of the fusion inputs to start spoofing on the other inputs (which have a higher weight in the fusion procedure). By doing so, the victim's vehicle will drift away from the road. Wan et al. [200], proposed a framework to identify DoS attacks against planning algorithms for autonomous vehicles. The proposed framework automatically finds driving situations that cause the vehicle to stop operating, by placing adversarial objects in the vehicle's surroundings.

5.5.4 Machine Learning security beyond CPS

The efficacy of Adversarial Machine Learning has been demonstrated in a wide range of applications outside Cyber-physical systems, for example, face [174] and voice recognition [211] and malware classification [208]. Biggio et al. [22] explored evasion attacks against PDF malware detection, Szegedy et al. proved the existence of adversarial examples for neural networks [185], while Goodfellow et al. [80] proposed the fast gradient method for adversarial perturbations, demonstrating its application in the image classification tasks. Carlini and Wagner [34] proposed attacks against adversarial machine learning defenses [154]. Attack on machine learning models is not limited to evasion, poisoning, and backdoors described in Section 2. Other attacks include Membership Inference attacks [176, 165], where the attackers infer the data points used for model training, and Model Stealing attacks [141], where the attacker's goal is to steal the model or approximate its behavior.

5.6 Organization of following chapters

In this introductory chapter to the evaluation of process-based anomaly detectors, we introduced the goals and approach to assess the security of anomaly detectors. In particular, we presented background information on the datasets for anomaly detection, proposed a taxonomy of process-based anomaly detectors proposed in prior work, and finally, introduced the system, attacker goals, and assumptions for concealment attacks. The content of this chapter will serve as common ground information provided in the next three chapters. The remainder of the Part is organized as follows.

In Chapter 6, we evaluate the security of prior-work reconstruction-based anomaly detectors under the proposed constrained attacker model. We show that replay attacks (commonly considered to be very strong) fail in concealing the anomalies when the attacker is subject to constraints. To address this, we propose white-box and black-box attacks that conceal anomalies by perturbing a subset of the sensor readings, leveraging learned physical constraints of the system.

In Chapter 7, we hypothesize that the detectors verify a combination of temporal, spatial, and statistical consistencies. We systematically analyze if the anomaly detectors are correctly modeling the consistency properties by proposing black-box generic concealment attacks. We apply our framework to a wide range of anomaly detectors. Our results demonstrate that the evaluated model-free detectors are susceptible to generic concealment attacks. For each evaluated detector, our framework shows which specific consistencies are verified. Moreover, we compare with the attacks from Chapter 6.

In Chapter 8, we are interested in understanding the minimal perturbation boundaries to evade ICS anomaly detectors. We propose white-box concealment attacks and generalize them to be adapted even if the target detection technique does not optimize a loss function. We design and implement a framework to perform our attacks and test it on several detectors from related work. Our results show that it is possible to completely evade a wide range of detectors (based on diverse detection techniques) while reducing the number of samples that need to be manipulated (compared to black-box concealment attacks proposed in Chapter 7).

6

Constrained Concealment Attacks

Constrained Concealment Attacks against Reconstruction-based Anomaly Detectors in Industrial Control Systems

6.1 Introduction

In the previous Chapter 5, we defined the concealment attack problem for anomaly detectors in cyber-physical systems. In this chapter, we start our evaluation of process-based anomaly detectors by *proposing and evaluating constrained concealment attacks against reconstruction-based anomaly detectors* (see Section 5.3.1).

In the context of cyber-physical systems, the adversarial examples obtained with a concealment attack must meet four requirements. **R1**: Due to the distributed nature of the system, the attacker is constrained to manipulate only a subset of features. **R2**: Adversarial examples must meet the temporal and spatial correlations expected from the observed physical processes [189, 91]. Particularly, adversarial examples must not introduce contextual anomalies (i.e., observations classified as abnormal only when viewed against other variables that characterize the behavior of the physical process [91]). **R3**: Most AML attacks in other domains target end-to-end Neural Network Classifiers instead of reconstruction-based classifiers. We realized that this requires the attacker to optimize the Mean Squared Error loss instead of optimizing the cross-entropy loss (Section 6.3.2). **R4**: To the best of our knowledge, previous work either assumes unlimited computational power to compute ideal perturbations [35] or assumes static systems that allow universal adversarial perturbations [135, 121]. For real-time attacks[1] on dynamic ICS, neither approach is feasible, and new solutions are required.

We formalize a detailed attacker model and evaluate different settings relating to attacker constraints, i.e., the number of features under the control of the attacker. The attacker leverages passive observation of system behavior to approximate how realistic examples should behave. Based on that, we consider a white box attacker that can leverage the knowledge of the system to perform iterative attacks on general reconstruction-based detectors. Moreover, we consider a black box attacker and show that it is possible to craft effective adversarial samples in milliseconds.

We summarize our main contributions as follows:

- Based on the attacker model proposed in the previous chapter we introduce constraints on the attacker motivated by real-world ICS.
- We show that replay attacks do not conceal anomalies when the attacker is constrained to manipulate less than 95% of the ICS features, due to physical correlations that are not exploited by such attacks.
- We evaluate our prior work concealment attacks for ICS process-based anomaly detectors [58] under the proposed constrained scenario. Our results show that such generated adversarial examples do not violate correlations (outperforming replay attacks in constrained scenarios). A white box attacker exploits the knowledge of the Anomaly Detection System launching an iterative attack based on a coordinate descent algorithm. A black box attacker without the knowledge of the Anomaly

[1] With real-time, we mean examples are crafted w.r.t. the current dynamic state of the system, in less time than the sampling rate (e.g., 10ms).

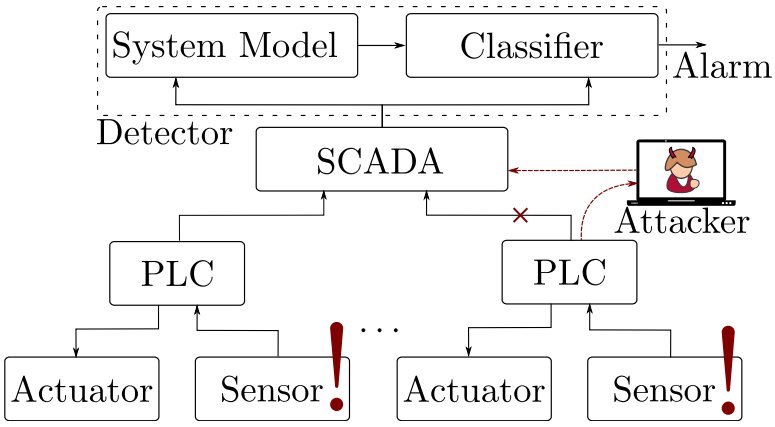


Figure 6.1: High level system and attacker model. The PLCs report sensor data about the anomalous process to the SCADA. The attacker can eavesdrop and manipulate a subset of the data provided to the SCADA. The reconstruction-based detector attempts to detect attacks based on a learned model of the system's benign operations.

Detection System uses learning-based attack leveraging adversarially trained autoencoders[2].

- We evaluate and discuss the proposed attacks, and compare their performance against replay attacks. The evaluation is conducted over the BATADAL and WADI ICS process datasets. The implementation is available at `https://github.com/scy-phy/ICS-Evasion-Attacks`.

The remainder of this chapter is structured as follows. We present the problem of adversarial learning attacks on reconstruction-based detectors in Section 6.2. The information on the attack methodology is introduced in Section 6.3. Implementation and evaluation is presented in Section 6.4. In Section 6.5, we summarize the findings presented in this chapter.

6.2 Concealment Attacks on Reconstruction-based Anomaly Detection

6.2.1 Concealment Constraints

Consistently with the system and attacker model presented in Section 5.4, we consider a system under attack consisting of several sensors and actuators connected to one or more PLCs, which are in turn connected to a SCADA system that gathers data from the PLCs. To assess the robustness of anomaly detectors, we use constraints on the number of sensors that can be manipulated by an attacker (see Section 5.4.3). Specifically, the concealment attack problem formulated in Equation 5.12, in the case of reconstruction-based anomaly detectors, can be formulated as the constrained optimization problem in

[2]Note: Not to be confused with Adversarial Autoencoders [127].

Table 6.1: Classification of our attacker models based on training data and features, Data tuple $(\mathcal{D}, \mathcal{X})$ and algorithm knowledge and parameters. Access:●=full,◐=partial. For all white box attacks, the Defense tuple is (f, w), for all black box attacks, $(\not{f}, \not{w})$.

Attacker's Constraints		$\mathcal{D}$	$\mathcal{X}$ Read	$\mathcal{X}$ Write
Unconstrained	§ 6.4.2	●	●	●
$\mathcal{X}$ Partially	§ 6.4.3	●	●	◐
$\mathcal{X}$ Fully	§ 6.4.3	●	◐	◐
$\mathcal{D}$	§ 6.4.3	◐	●	●

Equation 6.1.

$$\begin{aligned} \text{minimize} \quad & MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{x_i} - (x_i + \delta_i))^2 \\ \text{s.t.} \quad & \vec{\delta} \in \text{constraint space (Section 5.4.3)} \\ & \text{real-time constraints imposed by CPS} \\ & y(\vec{x} + \vec{\delta}) = \text{'safe'} \end{aligned} \tag{6.1}$$

As we will show, the performance of the attack degrades when lowering the number of channels under the attacker's control. Fully authenticated sensor signals would eventually prevent the attack from occurring at the process level but would impose costs on normal system operations. Our attack exploits sensor signals received by the detector, for this reason, the attack can be deployed at the upper layers of the industrial plant (see Purdue model Section 2.1.1.1). Software exploits on the machine running the detector or historian server could also offer the attack surface to mount our proposed concealment attacks (those alternative attacker models are not modeled here for the sake of simplicity).

Selection of Constrained Features The subset of features that can be modified by the attacker is highly use-case dependent (i.e., which link is attacked, which device was compromised). To demonstrate the generality of our findings, we explore two types of constraints: a best-case scenario and a topology-based scenario.

For the best-case scenario, we assume that the selection of the k out of n manipulated features can be made by the attacker to maximize the attack impact. This arguably represents a best-case scenario for the constrained attacker (i.e., an attacker constrained to features that happen to be relatively ideal for the attacker). For the second scenario, constraints are derived from the network topology. We assume that the attacker can compromise a single substation (or PLC) in the network, and the selection of k out of n features is based on which sensors are interconnected to the compromised substation.

6.2.2 Example Constraint Scenarios

We argue our *Constrained* and *Unconstrained* attacks introduced in the previous chapter (see Section 5.4.3) represent a realistic threat model in the ICS setting and fit the taxonomy of attacks in the AML literature. In particular, practical ICS are typically

composed of multiple stages, and each stage is controlled by a different PLC (i.e., different brands/models). Moreover, the ICS can be deployed in a physically distributed manner. For example, in the case of water distribution networks, pumping stations are typically located kilometers away from the water reservoir. In this heterogeneous setting, an attacker can either gain control over a limited set of resources as practically demonstrated in [71], or the whole plant (by compromising the network or central SCADA). We capture those different capabilities of the attacker in the following three scenarios:

- The *Unconstrained Attacker* can read and write any features arbitrarily, e.g., by compromising the SCADA system, as modeled by Mo et al. [134].
- The *Partially Constrained attacker* can read all traffic received by the SCADA system (for example by a passive wiretap [191] or by leveraging access control misconfigurations), but she is only able to spoof sensor readings from a specific substation, exploiting specific vulnerabilities of the substation or its protocol to the SCADA (e.g., lack of authentication).
- The *Fully Constrained attacker* relates to a scenario where an attacker compromised a specific substation, giving him read and write access to features from this substation only [3, 71].

Similar assumptions were also considered in the BATADAL dataset [189], where the attacker was assumed to perform constrained replay attacks to reduce the confidence of anomaly detectors. In our contribution, we perform a concealment attack in a systematic way to assess their impact on the anomaly detector. Table 6.1, provides an overview of the attacker's constraints considered in this chapter.

A similar intuition is also used in the FAIL attacker model [182] for AML. In particular, FAIL proposes to characterize attacker knowledge over four dimensions: **F**eature, **A**lgorithm, **I**nstance, **L**everage. While the first three have a counterpart in systematization by Biggio et al. [23], **L**everage stands for the subset of features that the adversary can modify (just like our constrained attacker). Thus, our *Unconstrained* and *Constrained* attacks represent attacks with full and limited **L**everage.

6.2.3 Framework for Attack Computation

Following the approach proposed in our prior work [58], we apply two methodologies for attack computation, white box and black box. In both scenarios, we assume the attacker can intercept and perturb a Constrained (or Unconstrained) set of sensor readings in real-time, and forward the perturbed samples to the SCADA system.

The white box attack applies an iterative methodology, in which the attacker uses a classifier oracle to perform a coordinate descent algorithm and minimize the MSE loss for the reconstruction-based anomaly detector (see Section 5.3.1), and consequently remove the anomaly from the process data.

The black box attack applies a learning based, in which a deep neural network is used to conceal sensor readings, without relying on the feedback from an oracle. The attacker

is required to train the neural network before launching the attack using eavesdropped PLC data from the process.

To avoid confusion, we point out explicitly that iterative attack can operate under the attacker white box assumption as it requires to query an oracle of the anomaly detector, while the learning based attack can operate under the black box assumption as no query access is required to compute the adversarial sensor readings.

6.3 Concealment Attack Methods

In this section, we summarize our prior work on concealment attacks for ICS detectors [58] considered in this chapter to apply the proposed constrained concealment methodology. We compare the concealment attack methodologies with replay attacks in the considered constraints scenario. Finally, we position our work w.r.t. state-of-the-art concealment attack methods.

6.3.1 Design of Concealment Attacks

We provide details on the iterative attack (white box knowledge) and the learning based approach (black box knowledge). We note that, while adversarial examples found using the iterative approach depend on the internal structure of the attacked anomaly detector, examples crafted through the learning based approach are independent of the addressed detection scheme (see Section 6.4.4).

6.3.1.1 Baseline: Replay Attack

In the replay attack setting (prior work, used here as baseline), the attacker does not know how detection is performed. In order to avoid detection, the attacker can replay sensor readings that have been recorded while no anomalies are occurring in the system. In particular, we assume that the attacker could record selected data occurring exactly n days before—i.e., if the concealment attack starts at 10 a.m., the attacker starts replaying data from 10 a.m. one day before.

6.3.1.2 Iterative Attack

In the iterative attack [58], we assume white box knowledge of the anomaly detection system (parameters and threshold) and the operational ranges of the model features. With this knowledge, the attacker has access to an oracle of the reconstruction-based anomaly detection system. The attacker can query the oracle with $\vec{x}$, as a result, the attacker gets the reconstruction error vector $\vec{e}$ (see Section 5.3.1) and can compute $\max_i \vec{e}$ to find the sensor reading r_i with the highest reconstruction error from $\vec{x}$.

To satisfy the constraint the concealment goal $\varepsilon(\vec{e'}) < \theta$ (i.e. $f(\vec{x_{adv}}) =$ 'safe' in Equation 5.12), the attacker performs a coordinate descent algorithm to decrease the reconstruction error related to r_i (As we rely on coordinate descent algorithm we do not use gradient estimation methods). At each iteration of the algorithm, a coordinate of the feature vector is modified until a solution is found or the budgets are consumed. The

search space for the coordinate adversarial value is the range observed during normal operating conditions.

Two computational budgets are put in place: *patience* and *budget*. If no lower reconstruction error is found by descending a coordinate, the algorithm tries descending other coordinates until the input is no longer optimized or the *patience* budget is consumed. *budget* is the maximum number of iterations for coordinate descent. After *budget* attempts without satisfying $\varepsilon(\vec{e}') < \theta$, the optimization terminates without solution.

6.3.1.3 Learning Based Attack

In the learning based setting [58], the black box attacker can not rely on the knowledge of the detection mechanism. For this reason, the attacker is assumed to rely exclusively on eavesdropped sensor readings to train an adversarial deep learning model, which is used to conceal the anomalies (referred to as *Autoencoder-based Generator*).

The Autoencoder-based Generator relies on a deep autoencoder to perturb the anomalous sensor readings. It is trained using eavesdropped normal data (without anomalies) and learns the non-anomalous patterns. When an anomaly occurs, the generator is used to conceal the anomaly. Later, the anomaly detection system will use the perturbed features for classification, causing misclassification of the system state. In practice, not all the features are perturbed by the attack but a subset of them (i.e., the anomalous features). After the prediction step by the generator, a post-processing step approximates to the closest integer all the categorical features, i.e., the actuator states.

The implementation of the Learning Based attack follows the architecture proposed in our prior work [58], with three hidden layers, with input and output dimensions equal to the number of sensors and actuators in the network, Mean Squared Error loss, and sigmoid activation function.

6.3.2 Positioning with respect to State-of-The-Art AML attacks

In this chapter, we consider attacks on a (prior work) reconstruction-based classifier. This represents a substantial difference from classifiers considered in AML attacks in other domains. In related work [153, 34, 124], attacks on end-to-end Neural Network classifiers are considered (i.e., those classifiers are trained to optimize the cross-entropy loss). In particular, target misclassification is achieved, diminishing the predicted probability of the true class in the output layer. Our problem setting is different: To evade the classifier, we need to diminish the residual between input and output. Our target Neural Networks are trained to optimize the Mean Squared Error loss. This can be achieved by reconstructing the sensor signal in a way that matches the learned physical properties of the ICS. Those differences motivated our white-box and black box approaches to evade Deep Learning-based Anomaly Detectors in ICS.

Model Robustness. Adversarial Robustness [124] is achieved by a) adversarial training b) classifier capacity. Following the definition, adversarial training is obtained by embedding adversarial examples in the training set. In our context, the Neural Network is trained to approximate the system behavior during normal operating conditions, with no samples for the 'anomalous' class. Thus, adversarial training here does not apply:

indeed, we cannot train the system to be resilient to adversarial attacks since samples from the class 'anomalous' are unknown to the defender.

6.4 Evaluation

In this section, we evaluate the concealment attacks. We evaluate the methodology on the BATADAL dataset and WADI dataset coming from a real industrial process (see Section 5.2.1). We start the evaluation targeting an Autoencoder-based detector and explore the performance of replay, iterative, and learning based attacks in constrained and unconstrained conditions. Then, we show that our learning based attack generalizes to other schemes based on LSTM and CNN.

6.4.1 Evaluation Setup

We evaluate the detection Recall (see Section 5.2.4) over datasets under original conditions (i.e., no concealment attacks), replay, iterative, and learning-based concealment attacks. When the anomaly detector is tuned, a higher Recall means that the anomaly detector is correctly retrieving anomalies. The attacker's goal is to reduce the Recall of the classifier (when 0 the attacker completely evaded the detector). Note that we launch our concealment attacks over the instances of anomalous data, i.e., data reporting ground truth 'anomalous'.

Both iterative and learning based attacks are implemented using Python 3.7.1; neural networks are implemented and trained using Keras 2.3.1 with TensorFlow 1.11.0 backend. Experiments use a laptop equipped with Intel i7-7500U CPU, 16GB of RAM, and NVIDIA GeForce 940MX GPU 4GB.

Target Detector. As reference implementation, we use the general Autoencoder-based anomaly detector framework proposed in [188] and available as open source [187]. Moreover, we explore the transferability of our black box attack between different Deep Architectures (DA). In particular, we tested Long Short Term Memory (LSTM) [94] as proposed in [77] and Convolutional Neural Networks (CNN) [113] as proposed in [112].

Training of Attack Detector. We train the autoencoder based anomaly detector proposed by Taormina et al. [188] over the BATADAL and WADI datasets.

After hyperparameter tuning on the BATADAL dataset, we selected the window parameter to 3 quarter of hours. The resulting performance of the model is $Accuracy = 0.93$, $Precision = 0.90$, $Recall = 0.60$, $FPR = 0.01$.

For the WADI dataset, we tuned the window parameter to 60 seconds. Obtaining $Accuracy = 0.97$, $Precision = 0.77$, $Recall = 0.68$, $FPR = 0.01$.

Results are in line with the current state-of-the-art detection over the BATADAL dataset [188] and WADI [62].

Replay attack. In this attack, the attacker replays while a physical manipulation is occurring on the system, using the sensor readings as recorded at the same hour S days before (assuming that process operations are often periodic within 24h). S is chosen to let the replay contain only normal operations data. For example, given a physical manipulation that lasts 50 hours, we replay sensor readings as happened 72 hours earlier.

Table 6.2: Detector Recall (BATADAL (B) and WADI (W) datasets), before and after unconstrained concealment attacks. The column 'Original' refers to the detection Recall over the data without concealment; 'Replay', reports the Recall after replay attack, while 'Iterative' and 'Learning based' columns report the Recall after our proposed adversarial concealment attacks.

	Detection Recall			
Data	Original	Replay	Iterative	Learning based
B	0.60	0	0.14	0.14
W	0.68	0.07	0.07	0.31

Iterative attack. The attacker manipulates variables required to find a solution (according to the two stopping criteria introduced in Section 6.3.1.2 and constraints over modifiable sensor readings). Following the tuning proposed in our prior work [58], for the BATADAL dataset, we tuned the two stopping criteria via grid search to guarantee a trade-off between the decrease of detection accuracy and computational time. Specifically we selected *patience* = 15 and the *budget* = 200. For WADI dataset, the iterative parameters (following the same rational as in BATADAL case) we choose are *patience* = 40 and the *budget* = 300.

Learning based attack. For the learning based attack, the attacker uses an autoencoder (AE) as the generator and sends predicted readings to the SCADA. According to the attacker's constraints, we train an autoencoder over the readable features. We used sigmoid as activation function, Gorlot initialization [74] as weights initializer and mean squared error as loss function. Moreover, we split the data in train $\frac{2}{3}$ and validation $\frac{1}{3}$, use early stopping [54] to avoid overfitting and reduce learning rate on plateaus [159]. Depending on the constrained scenario (i.e., the features that the attacker can read $\mathcal{X}$ or the amount of data that she spoofed $\hat{\mathcal{D}}$), the adversarially trained autoencoder has a different number of input neurons. Given n as input/output dimension, the autoencoder is composed of 3 hidden-layers with respectively $2n$, $4n$, $2n$ neurons. To perform and evaluate the learning based attacks, we trained 83 models with BATADAL data and 63 for WADI. In the unconstrained case, for the BATADAL dataset (43 variables), we train an autoencoder with 64 and 128 units for the first/third and second hidden layers, respectively, training requires 18 epochs (2 seconds/epoch), for a total of 36 seconds to train the model; for the WADI dataset (82 variables), we use 128 and 256 units, training required 7 epochs (64 seconds/epoch), for a total of 488 seconds to train the model.

6.4.2 Unconstrained Concealment Attack

In this experiment, we assume the Unconstrained attacker $(\mathcal{D}, \mathcal{X})$ that is able to read and control all the reported sensor readings, in the White box (f, w) and Black Box $(\not{f}, \not{w})$ scenarios. We discuss the results of our evaluation of the detector for both datasets in several scenarios (see Table 6.2). We evaluated the performance of our concealment attacks over the time steps with ground truth 'anomalous' labels only, i.e., we exclude normal operation data time steps from the computation of Recall for this attack evaluation.

Table 6.3: Average required time (in seconds) to manipulate sensor readings. 'Replay' column is empty as replay attacks do not require computation. 'Iterative' and 'Learning based' columns report the mean and std deviation required to compute the manipulation sensor readings at a given time step.

	Computational time, mean($\mu_{\bar{x}}$) and std($\sigma_{\bar{x}}$)				
	Replay	Iterative		Learning based	
Data		$\mu_{\bar{x}}$[s]	$\sigma_{\bar{x}}$	$\mu_{\bar{x}}$[s]	$\sigma_{\bar{x}}$
B	-	2.28	2.46	0.002	0.005
W	-	0.60	0.41	0.005	0.002

The first row of Table 6.2 reports the average results obtained with the three different attack strategies. In this setting, the replay attack is giving 0 Recall over the replayed sensor readings. This means that when the attacker can manipulate all the sensor readings, the anomaly detector is no more able to spot the attack occurring over the physical process. Considering the iterative and learning based approaches, we notice that the Recall is 0.14, this represents a significant drop in detector performance, but not as effective as the replay of all sensors.

The second row of Table 6.2 refers to concealment attacks over the WADI dataset. The result over this dataset shows that the replay attack can hide the anomaly occurring over the CPS. The performance of the iterative equals the one of the replay attack. Finally, the learning based approach is underperforming the other methods. Despite this, the detector's Recall reduces more than 50% after learning based manipulation.

Computational Time. Table 6.3 reports the average time required to compute the adversarial examples. In contrast to iterative and learning based, the replay attack does not require computation. The iterative approach requires an amount of time that depends on the algorithm computational budgets. The black box approach requires a constant amount of time since it consists of a neural network prediction. Given our real-time constraints of adversarial examples computation (i.e., target time within milliseconds), we can conclude that learning based approach easily meets the requirements. In the BATADAL case (where the sampling time is 15 minutes), we do not require more than $2ms$ on average to compute an adversarial example. In the WADI case (where sampling time is 1 second), on average, we do not require more than $5ms$ to compute an adversarial example. The iterative attack is slower, but on average, still below the sampling intervals.

Summary of Unconstrained Attacks findings. When the attacker is free to manipulate all the sensor readings, results show that replay attacks hide anomalies occurring over the physical process. First, a replay attack does not require computation to find the manipulated set of sensor readings; second, the attacker does not need to be aware of the detection mechanism; and third, the considered anomaly detector Recall goes to zero since the replayed sensor readings do not contain (additional) anomalies. White box, even though it achieves valuable results, requires computation, and the attacker needs to be omniscient w.r.t. the defense mechanism. We note that the learning based attack can decrease the detector's Recall without having access to detector's

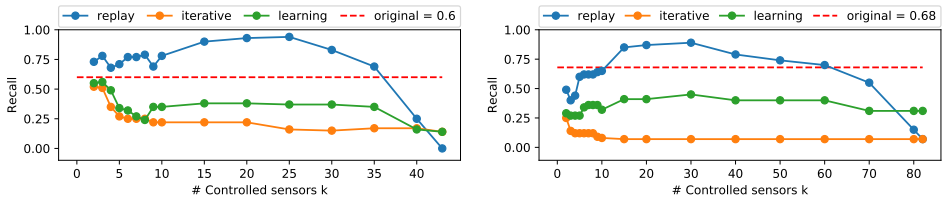
oracle, with low computational effort (after training) and the same knowledge w.r.t. the attacked model as the replay attack.

6.4.3 Constrained Concealment Attack

In the previous subsection, we found that full replay attacks can be a powerful and low-cost way to evade anomaly detectors if all features can be replayed. In this section, we demonstrate the impact of constraints on the attacker, e.g., if the attacker can only control a subset of the reported sensor values. Specifically, we perform Partially, Fully, and data-constrained attacks as modeled in Table 6.1 and show how our proposed iterative and learning based outperform replay attacks.

Partially Feature-Constrained attack, $(\mathcal{D}, \hat{\mathcal{X}})$. Figure 6.2 reports the average result of the constrained attacks over BATADAL and WADI datasets with an *best-case* selection of constraints. In order to study the impact of this best-case constraints, we selected k features for every attack that can be modified. Then we studied how replay, iterative, and learning based attacks perform when these constraint are applied. We defined the constrains as follows: starting from the results of iterative and learning based attacks we determined the k features that were changed most frequently (over the course of each attack). The intuition behind this is as follows: features that are modified most often in the unconstrained case are assumed to have the highest impact on the performance of the attack. We are assuming a best-case scenario for the attacker, in which she was able to choose the k our of n features to maximise her efficiency. Then, we created 11 sets of k features that can be modified by the attacker (with different counts k of features to be manipulated, with the maximal number determined by the dataset used). In the case of iterative and learning based attack, we limited the adversarial example exploration to the k features extracted for the considered approach. Effectively, that implies that we only used the allowed k features out of the learning based model, while the iterative model was able to apply modifications to the k allowed features that would minimize the detector accuracy. In the case of replay attack we applied the same replay strategy introduced before but we replayed only the selected k features extracted from the iterative approach. We note that this choice (replay the features extracted from the iterative approach) was made to reflect worst case scenario, i.e., an attacker that is able to replay exactly the k features that an iterative attacker would replay.

In the case of the BATADAL dataset, we note that the replay attack does not cope well with constraints. Since the anomaly detector can spot the presence of contextual anomalies, the replay of only k features results in alarms, with an average detection Recall higher than in the benign case (i.e., no replay of sensors applied), the value decreases when 40 out of 43 sensors are replayed. In the case of iterative and learning based attacks, we can notice that the detection Recall is always lower than the original Recall. In the iterative case, Recall decreases with the number of features that can be modified. Learning-based attack Recall is not monotonically decreasing with the number of features that can be modified. Specific constraint sets better match the physical rules learned by the detector and allow the creation of more effective adversarial examples. In the case of WADI, we can observe that the replay attack can diminish the detector's Recall, especially when the attacker manipulates 3 or more features. The



(a) BATADAL (b) WADI

Figure 6.2: Impact of Partially Constrained attacker (best-case scenario), comparison between replay attack and our proposed concealment attacks. In the constrained scenario, we notice that replay attack performs bad increasing detector's Recall and raising more alarms than the original data without concealment. This is due to contextual anomalies introduced as part of large-scale replay. Both learning based and iterative approaches outperform the replay attack as they do not introduce contextual anomalies and reduce detector's Recall manipulating few features.

iterative based attack can achieve the same Recall as if in the Unconstrained Attack case when manipulating 15 out of 82 features. In the case of learning based attack, results show that for 3 manipulated (best-case) features, the attack performs slightly better than in the unconstrained case.

In the case of *topology-based* constraints, the attacker controls the sensors connected to 1 PLC in the network. We found that in the BATADAL case, Recall is reduced to 0.36 with the replay attack and 0.34 with the iterative and learning-based attack. In the WADI case, Recall increases to 0.64 with the replay attack while it is reduced to 0.12 in the iterative attack and 0.36 in the learning-based attack. In addition, in this case the iterative and learning-based approach overcomes the limitations of constrained replay attacks, especially in the case of the WADI dataset (Table 6.4 reports the numerical Recall scores found in the different constrained settings).

Fully Feature-Constrained Attacker, $(\mathcal{D}, \hat{\mathcal{X}})$. In the case of the fully constrained attacker, Replay and Iterative attack approach do not change, since those two methods do not exploit correlations among features to output the perturbations. The learning-based attack is the only one affected by these constraints, i.e., the adversarial network is trained on the constrained set of sensor readings. We launched this attack with the topology based constraints, and we found that also in this case, when the attacker gains control of a PLC over the network, the detection Recall can be compromised by the attacker. In the BATADAL case, Recall is reduced to 0.39. In the WADI case is reduced to 0.45.

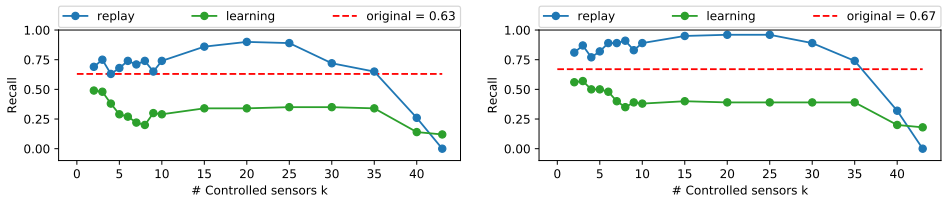
Data Constrained Attacker, $(\hat{\mathcal{D}}, \mathcal{X})$. In the case of the black box attack, we have investigated the impact of a fraction of training data available to the attacker. For BATADAL, the resulting mean detection Recall ranges from 0.14 for 100% of $\hat{\mathcal{D}}$ available for AE training to 0.22 for 5% of $\hat{\mathcal{D}}$ available. For WADI, we found that the attacker can leverage 5% of normal operations (i.e., only 16 hours) to decrease detector Recall to 0.24.

Summary of Constrained Attacks findings. Our results demonstrate that replay attacks perform worse if a limited set of sensors can be manipulated. In particular, if

Table 6.4: Impact of Partially Constrained attacker (best-case scenario and topology based scenario). By decreasing the number of features that the attacker can control, we notice that replay attack performance decreases drastically. This is due to contextual anomalies introduced as part of large-scale replay, while both black and iterative approaches avoid this problem.

	Original		Recall vs. # of Controlled sensors k (43 features BATADAL)																Topology
Data	Recall	Experiment	43	40	35	30	25	20	15	10	9	8	7	6	5	4	3	2	1 PLC
B	0.60	replay	0	0.25	0.69	0.83	0.94	0.93	0.9	0.78	0.69	0.79	0.77	0.77	0.71	0.68	0.78	0.73	0.36
		iterative	0.14	0.17	0.17	0.15	0.16	0.22	0.22	0.22	0.22	0.25	0.25	0.25	0.27	0.35	0.51	0.52	0.34
		learning	0.14	0.16	0.35	0.37	0.37	0.38	0.38	0.35	0.35	0.24	0.27	0.32	0.34	0.49	0.56	0.55	0.34

	Original		Recall vs. # of Controlled sensors k (82 features WADI)																Topology
Data	Recall	Experiment	82	80	70	60	50	40	30	20	15	10	9	8	7	6	5	4	3	2	1 PLC
W	0.68	replay	0.07	0.15	0.55	0.7	0.74	0.79	0.89	0.87	0.85	0.65	0.64	0.62	0.62	0.62	0.6	0.44	0.4	0.49	0.64
		iterative	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.08	0.09	0.12	0.12	0.12	0.12	0.12	0.14	0.25	0.12
		learning	0.31	0.31	0.31	0.4	0.4	0.4	0.45	0.41	0.41	0.32	0.36	0.36	0.36	0.34	0.27	0.27	0.27	0.29	0.36



(a) LSTM (b) CNN

Figure 6.3: Generizability of our proposed Learning based attack compared with replay attack. Attack to LSTM (a) and CNN (b) based defenses on BATADAL dataset.

the replay attack is constrained to manipulate less the 95% of the features the detector's Recall *increases* due to contextual anomalies that are created. For our iterative and learning based approaches, this effect does not occur, as the two attacks reduce the detector's Recall without introducing contextual anomalies. In the WADI, the attacker that controls 4% of the features succeeds in the evasion using the iterative attack. In addition, an attacker that collects a little amount of data $\hat{\mathcal{D}}$ can train the adversarial autoencoder and perform the black box attack.

6.4.4 Generalizability of Learning Based Attack

In this section, we evaluate the generalizability of our proposed learning based attack. We consider different reconstruction-based anomaly detectors trained on BATADAL dataset and apply the concealment attack computed with the adversarially trained autoencoder as proposed in our learning based attack. Our target Deep Architecture is an LSTM reconstruction-based anomaly detector as proposed in [77] and a CNN reconstruction-based anomaly detector as proposed in [112]. Since those detectors are not available as open source, we implemented their architecture with Keras following the details found in the paper. The LSTM detector was trained to minimize the MSE loss. The network takes 8 timesteps of the multivariate time series as input (input size $8x43$). The input is processed by one LSTM layer with output size 43, followed by a fully connected layer with 43 neurons as output dimension. Performance of the LSTM

Table 6.5: Generizability of Learning based attack. Attack to LSTM and CNN based defenses on BATADAL dataset.

	Original		Recall vs. # of Controlled sensors k (43 features BATADAL)														
Data	Recall	Experiment	43	40	35	30	25	20	15	10	9	8	7	6	5	4	3	2
LSTM	0.63	replay	0	0.26	0.65	0.72	0.89	0.9	0.86	0.74	0.65	0.74	0.71	0.74	0.68	0.63	0.75	0.69
		learning	0.12	0.14	0.34	0.35	0.35	0.34	0.34	0.29	0.3	0.2	0.22	0.27	0.29	0.38	0.48	0.49
CNN	0.67	replay	0	0.32	0.74	0.89	0.96	0.96	0.95	0.89	0.83	0.91	0.89	0.89	0.82	0.77	0.87	0.81
		learning	0.18	0.2	0.39	0.39	0.39	0.39	0.4	0.38	0.39	0.35	0.4	0.48	0.5	0.5	0.57	0.56

based anomaly detector resulted in $Accuracy = 0.94$, $Precision = 0.89$, $Recall = 0.63$, $FPR = 0.01$. The CNN detector was trained to minimize the MSE loss. The network takes 2 timesteps of the multivariate time series as input (input size $2x43$). The input is processed by three stacked 1D convolutional layers (respectively with 64, 128 and 256 neurons), each convolutional layer is followed by a 1D Max Pooling Layer layer. The last Pooling layer is followed by a flatten and a dropout layer. Dropout layer connects to the fully connected output layer with dimension 43. CNN based performance after training resulted in $Accuracy = 0.95$, $Precision = 0.90$, $Recall = 0.67$, $FPR = 0.01$.

Results in Figure 6.3 and Table 6.5 show how the detection Recall diminishes when targeted with replay and learning based attacks. In particular, a replay attack can evade detection only in the case in which at least 40 out of 43 sensors are controlled by the attacker (following previous results). Results over learning based attack, despite the different architectures for the offense (Autoencoder) and defense (LSTM and CNN), show that the learning based concealment attack is transferable. In particular, the LSTM architecture appears more vulnerable to concealment attacks, since the learning based attack is achieving higher concealment efficacy than AE and CNN based defenses. Concealment efficacy over CNN defense is comparable to Autoencoder defense, notwithstanding the original Recall of CNN detector is higher than the other considered defense. These results allow us to conclude that the proposed learning based attack efficacy is independent from the Reconstruction-based anomaly detector.

6.5 Conclusions on Constrained Concealment Attacks

In this chapter, based on our proposed attacker model, we presented the constrained concealment attacks on reconstruction-based anomaly detectors in the context of Industrial Control Systems. We argued that such attacks present four unique challenges, (R1: Mean-Squared Error loss, R2: Temporal and spatial correlations, R3: Real-time evasion, and R4: Feature-constrained attacker) and addressed them proposing iterative and learning based constrained attacks to evade the anomaly detectors.

Using data from two water distribution systems, we demonstrated that our attacks evade the reconstruction-based anomaly detectors, and outperform replay attacks when the attacker is constrained to control less than 95% of the features, as the correlations among features are not modeled in a constrained replay attack. Our results demonstrate that the proposed learning based attack achieves successful concealment without knowledge of the targeted Reconstruction-based anomaly detector (only using normal operational data), without knowledge of the physical model equations, and is computationally cheap (after training).

Moreover, the proposed attacks allow attackers to perform constrained concealment attacks on dynamic systems in real-time. In prior work, manipulations are usually performed offline against a dataset or assume that the data to be manipulated can be precisely predicted. Our results show that reconstruction-based attack detectors proposed in prior work are vulnerable to manipulation despite the unique challenges in this setting, and such attacks need to be considered when designing future attack detection schemes. Implementation is available at our online repository `https://github.com/scy-phy/ICS-Evasion-Attacks`.

In the next chapter, we will explore further the concept of correlations in the process features and propose a framework to assess the resilience of process-based detectors based on the process correlations.

7

Generic Concealment Attacks

Assessing Model-free Anomaly Detection in Industrial Control Systems Against Generic Concealment Attacks

7.1 Introduction

In the previous Chapter 6, we identified that constrained replay attacks are detected because they break the correlations among process features. In this chapter, we focus on the process correlations to characterize which properties of the physical process are learned by anomaly detectors.

Training and evaluation of anomaly detectors require operational and attack data [189, 101, 129, 76]. Due to the difficulty in creating realistic datasets (requiring practical testbeds with implementations of attacks, or detailed cyber-physical simulators), the diversity of attacks represented in them is limited. In particular, the major class of concealment attacks is insufficiently presented in the datasets (this phenomenon is referred to as sampling bias in related work [14]). As a result, the evaluation would cover a subset of threats the systems are designed to protect against. This leads to unexpected vulnerabilities as shown for Deep-Learning detectors [184].

In this chapter, to evaluate the security of process-based anomaly detectors, we introduce *generic concealment attacks* that verify how well a target anomaly detector abstracts the physical process properties to detect anomalies in the system. Specifically, we address four research questions. **RQ7.1**: *Are there fundamental properties that are checked by the anomaly detectors to identify anomalies?* **RQ7.2**: *How resilient are different model-free anomaly detection approaches against generic concealment attacks?* **RQ7.3**: *How do generic concealment attacks compare against prior work that targeted neural network-based detectors?* and **RQ7.4**: *How can we constructively build an anomaly detector starting from the identified fundamental properties?*

To answer **RQ7.1**, we show that manipulations of the physical process will violate the consistency of the system. In particular, we differentiate between spatial, temporal, and statistical consistency. We then consider three classes of generic concealment attacks that each specifically violate one of those consistencies. For each attack primitive, the goal of the attacker is to evade the detection of anomalous system states by manipulating selected sensor values.

To answer **RQ7.2**, we propose a framework to test process-based anomaly detection systems against our three distinct concealment attack primitives. We apply this framework to six prior work model-free anomaly detection systems. Namely, AR models [85], PASAD [9], SFIG [62] and Autoencoders [77, 188, 112].

To answer **RQ7.3**, we also evaluate the six anomaly detection systems against attacks from prior work that specifically targeted neural network-based detectors [P1].

To answer **RQ7.4**, we demonstrate how to construct an ensemble detector that reliably detects process anomalies and the evaluated concealment attacks.

We summarize our main contributions as follows:

- We provide the first systematic analysis of model-free process-based anomaly detectors.
- We introduce the concept of spatial, temporal, and statistical consistency to describe properties implicitly verified by the detectors, and how they relate to a

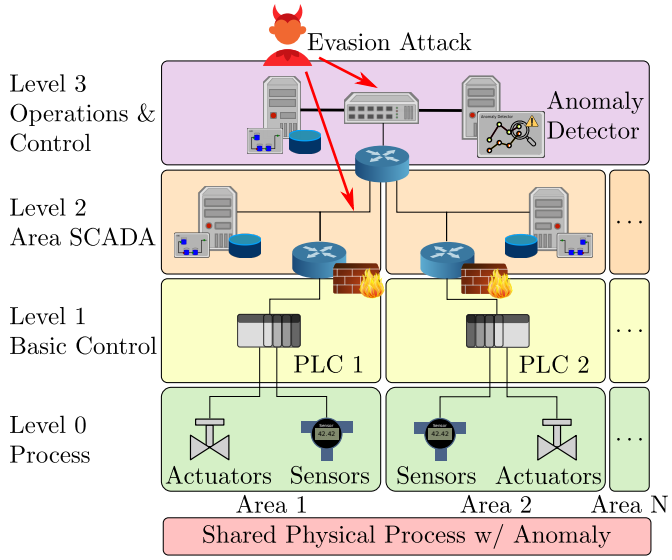


Figure 7.1: An Industrial Control System organized according to the Purdue Enterprise Reference Architecture (PERA). An attacker tampers with the physical process (Level 0), and performs a concealment attack to remain stealthy (Level 3).

state-space representation of the process.

- We practically implement three generic concealment attacks that are not so far represented in related public datasets, and demonstrate the attacks' efficacy against six model-free anomaly detectors from literature [85, 9, 62, 188, 112, 77]. We show that (surprisingly) even very basic attacks are effective (e.g., leading to a Recall of 0.0). We also evaluate attacks presented in Chapter 6 ([P1]) against the detectors.
- We propose an ensemble-based detector resilient against all evaluated concealment attack primitives, while also performing well in detecting the *original process anomalies in the dataset*.

Our implementation of the concealment attacks (extending public datasets), and our ensemble countermeasure are publicly available at `https://github.com/scy-phy/ICS_Generic_Concealment_Attacks`.

7.2 Classification of ICS Process Manipulation

In this section, we provide a taxonomy for attacks against ICS that are represented into datasets. We differentiate attacks between white-box (attacker knows all details of detectors and/or process) and black-box attacks (attacker does not have details on detector and/or process). In this chapter, we focus on generic black-box attacks. White-box attacks were explored in prior work [P1, 195]. Several black-box attacks on ICS processes have been proposed in prior work. Unfortunately, there is no systematic classification of the attacks so far. In this chapter, we classify those attacks as follows (see Table 7.1). Attacks are either of type *Overt or Evasion* (i.e., *does the attacker attempt*

Table 7.1: Taxonomy of black-box attacks in CPS. For each attack we report if it was represented in the dataset (●) or not (○) and in which research paper it was considered. † indicates the new attack that we have identified for this chapter.

	L0/ L2	Attack	Batadal	SWaT[129]	WaDI[101]
Overt		Random Manipul. [6, 7]	●	●	●
		Linear Transform. [6, 7]	●	●	●
		Stale Data L0 [116]	○	○	○
Evasion	Stealthy (L0)	Boiling Frog	●	●	●
	Concealment	Full Replay [134, P1]	○	○	○
		Constr. Replay [6, 188, 189, P1]	●	○	○
		Random Replay†	○	○	○
		Stale Data L2	○	○	○
		Learning-based [P1]	●	○	●

to hide from a detector?). We further differentiate Evasion attacks as either *stealthy or concealment*: Manipulated sensor values are identical at all receivers (stealthy), or they can differ between the receivers in the process and the SCADA/detector (concealment). For example, hidden manipulation at L0 requires stealthy attacks – as both the process and the SCADA observe the manipulation. In contrast, separate manipulation at L0 and L2 allow concealment attacks towards the SCADA/detector.

Overt Attacks. In this category we cover 'Random Manipulation' attacks in which the attacker changes a sensor/actuator to a different value without engineering of the spoofed value [6, 7]. There can be several reasons for this manipulation, for example, the attacker plans to destabilize the physical process or break some industrial equipment. The second category is the 'Linear Transformation' attacks [6, 7], where the attacker adds a constant offset or sets the sensor reading to a specific critical value to destabilize the physical process and cause the wrong control decision. The last example in this category is the 'Stale Data' attack [116], where the attacker launches a DoS attack on L0 industrial communications, leading to receivers falling back to using the last received value, which causes the system to perform erroneous control actions.

Evasion Attacks. In this category we consider two subcategories. The first category contains stealthy attacks, in particular, the 'Boiling Frog' attack in which the attacker manipulates the process characteristics slowly to drive the system to unsafe states without triggering an alarm. Examples of these attacks can be found in all the datasets that we analyzed in Table 7.1. The second category is concealment attacks, which attempt to hide anomalous sensor readings to the SCADA/detector by reporting the erroneous sensor readings. The first representative attack is the 'Replay' attack [134], where the attacker replays recorded sensor readings occurred in the past. If the attacker can replay all the sensor readings we have the 'Full Replay', otherwise the 'Constrained Replay' [189]. Replay attacks are very challenging to detect as the sensor readings do not present anomalies. The second category we have 'Random Replay' attack that we introduce in this chapter, see details in Section 7.3. The third category is the 'Stale Data' attack, where the attacker performs the DoS to L2 devices to make

the SCADA blind w.r.t. what happens on the physical process. Finally, there are Learning-based attacks [P1], in contrast to the other attacks this attack requires real-time calculations to compute the spoofing samples and was proposed to specifically evade reconstruction-based detectors.

From Table 7.1, it is clear that the most widely used datasets in this domain do not provide examples of all attack classes, and thus so far not been considered for the evaluation of existing model-free detectors.

7.3 Research Questions and Assumptions

7.3.1 Research Questions and Challenges

Limitations of Prior Work. In prior work, process-based anomaly detection schemes have been proposed to detect the effects of adversarial manipulations of an industrial process. In particular, model-free anomaly detection schemes aim to achieve this goal without explicit knowledge of the physical process. Unfortunately, the datasets used to train those schemes did not contain the important class of concealment attacks (popularized by Stuxnet [202]), in which the attacker aims to hide anomalies by manipulating the reported sensor data (see Section 7.2). In addition, while it was observed that different detectors appeared to be more suitable towards detecting specific attacks, no systematic analysis of the abstract data properties verified by the detectors was performed. Even when 'temporal and spatial correlations' were mentioned, they were not further analysed or specified [P1].

Research Questions. To address this gap, in this chapter, we address the four research questions presented in the introduction. Overall, answering the questions will allow us to a) provide guidance toward the design of future process-aware anomaly detectors, b) provide more complete datasets for detector design and evaluation, and c) better understand the threat of generic concealment attacks that target many different detector designs at once.

Challenges. Investigating the aforementioned research questions is challenging for a number of reasons. *i)* First, concealment attacks have not been systematically investigated in prior work. As there is no prior exhaustive enumeration of attack types, it is also unclear if datasets used to train and test detectors are comprehensive in the types of attacks they cover, and what types of attacks are successfully detected by the resulting schemes. In addition, prior work usually only presents the high-level concepts behind attacks and does not provide reference implementations or datasets featuring the attacks. *ii)* Second, prior work detectors are often difficult to replicate, as private datasets are used, source code is not shared (including, e.g, hyper-parameters), or custom evaluation criteria are used (see Section 5.3). That implies that any systematic investigation of multiple detectors will require the design and implementation of a common framework that allows to evaluate several detectors over a set of common datasets, using identical performance metrics.

7.3.2 System and Attacker Model

System Model. The considered system is an Industrial Control System organized following the Purdue architecture and equipped with an anomaly detection system, consistent with the description provided in Chapter 5, Section 5.4.

No Process and Detector Knowledge & Stealth. In contrast to assumptions in prior work [9, 62, 195, 7], our black-box attacker is weaker as they do not know process physics, i.e., they are unaware of the physical properties of the system and the impact they have on the multivariate temporal series generated by sensor readings (spatial consistency, temporal evolution, and statistical properties). Prior work discussed white-box attacks [195], and has already shown that if the attacker has detailed white-box knowledge, the best a countermeasure can do is to prolong the time until the attack succeeds (or reduce the impact, but not fully prevent it). In this chapter (see Section 7.5), we show that even black-box attacks (for which such detailed knowledge is not required) are successful for the evaluated prior work detectors. Moreover, the attacker does not know the inner workings of the anomaly detector and cannot access its parameters and detection scores. Despite these constraints, the attacker aims to hide an anomaly in the physical process from the anomaly detector, which would instead lead to an alarm by the detector. The attacker attempts to perform such a concealment attack to increase the overall damage caused over time, and cover the attacker's traces (see Stuxnet [202]).

With respect to the attacker space introduced in Chapter 5.4.3, in this Chapter we consider Unconstrained black box attackers ($\mathcal{D}, \mathcal{X}, \not{f}, \not{\mathcal{M}}$) and Feature fully-constrained black box ($\mathcal{D}, \hat{\mathcal{X}}, \not{f}, \not{\mathcal{M}}$) attackers.

Capabilities. As in the prior work detectors summarized in Section 5.3, our attacker is assumed to be capable of i) dropping traffic towards the detector, ii) manipulate traffic towards the detector, and iii) eavesdrop traffic towards the detector. We will also investigate attacks that do not even require eavesdropping or manipulation of traffic (i.e., stale data attacks). We note that eavesdropping and manipulating traffic can be achieved in many ways due to missing security in industrial protocols, e.g., wireless jamming, packet dropping by attackers controlling forwarding devices, routing and ARP-based attacks, etc. [151, 161, 196]. According to prior work [82, 166, 106], the detailed process knowledge of complex systems is commonly not assumed, and obtaining it is challenging (if not impossible) for many attackers. We also note that the analyzed model-free detectors were proposed for settings where even the system operators have insufficient process knowledge to simulate the process fully.

Attack duration. The Attacker's goal is to conceal ongoing anomalies on the system, so we assume that the attacker launches the concealment while there is an ongoing anomaly on the system. As we rely on datasets with labeled anomalies, the duration of a concealment attack is determined by the 'anomalous' label in the dataset.

Manipulation Constraints. As the physical process observed by the anomaly detector can encompass large areas with multiple sites (e.g., networks, substations, plants, etc.), we will also evaluate *constraints* on the attacker. For example, the attacker might only be able to manipulate a subset of the sensors as seen by the detector.

7.4 Consistencies & Their Verification

While general properties of the physical process sensor data are often (implicitly or explicitly claimed to be) verified by ICS and in general CPS anomaly detection systems, no consistent understanding or investigation of those properties was proposed in CPS anomaly detection works. To address this gap (**RQ7.1**), we introduce three types of consistency that hold in the process sensor data by leveraging the notion of State-space representation to describe the deterministic behavior that characterizes a CPS. We call the consistencies *temporal*, *spatial*, and *statistical* consistency. Based on the identified consistency properties we then provide a high-level description of the three generic concealment attacks that were not part of prior work evaluation.

7.4.1 State-space representation

Physical systems behavior is deterministically modeled with the so-called State-space representation [39], Equation 7.1 represents a discrete-time system. This representation combines the input, the system state, and the physical properties of the system to derive the evolution of the state and the output of the system. This deterministic representation captures the relation between the system's physical properties and allows its control.

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + v_k \\ y_k = Cx_k + Du_k + w_k \end{cases} \tag{7.1}$$

Where $k := kT$ and T is the sampling time. $x_k \in \mathbb{R}^n$ represents the state of the system, which is defined as the set of variables (directly or indirectly measurable) that characterize the physical system at a given time. This set of variables defines a Euclidean-space i.e., the State-space, and the state of the system at time k, i.e., x_k is a vector in the State-space. $u_k \in \mathbb{R}^p$ represents the input (or control vector) to the system, it influences the state of the system x_k and its output y_k. In a feedback control loop, u_k is the output of the controller. $y_k \in \mathbb{R}^q$ represents the output of the system, and it can be measured with sensors and it is influenced by the input u_k and the state x_k. $v_k \sim WN(0, V) \in \mathbb{R}^n$ and $w_k \sim WN(0, W) \in \mathbb{R}^q$ are white noise disturbances and $V \in \mathbb{R}^{p \times p}$, $W \in \mathbb{R}^{q \times q}$ are the noise variance matrices. $A \in \mathbb{R}^{n \times n}$ is the state matrix, it contains the coefficients of the physical relationship between the state x_k and its update x_{k+1}. $B \in \mathbb{R}^{n \times p}$ is the input matrix, it contains the coefficients of the physical relation between the system input u_k and the state update x_{k+1}. $C \in \mathbb{R}^{q \times n}$ is the output matrix, it contains the coefficients of the relation between the state x_k and the measured output y_k. $D \in \mathbb{R}^{q \times p}$ is the feed-through matrix, it contains the coefficients of the dependence between u_k and y_k.

Anomalies. In a time-invariant system, A, B, C, D are constants. In case of an attack on the physical process, at least one of those matrices is changed (as the matrices represent the physical process). In other words, the changed process becomes inconsistent with the normal process. We introduce in detail three different types of inconsistencies: spatial, temporal, and statistical.

7.4.2 Spatial Consistency

Spatial consistency refers to the correlation among quantities measured at the same instant in the physical process (referred to as attribute correlations by Illiano et al. [98]). This correlation depends on the physical process and control action. Considering the state-space representation of a Linear Time-Invariant system, the output y_k is observed by a set of sensors. Given the system state x_k, (i.e., $Ax_{k-1} + Bu_{k-1}$) and input u_k of the physical system, the values of the output features are correlated according to the equation:

$$\hat{y}_k = Cx_k + Du_k \tag{7.2}$$

An anomaly detector should correctly exploit those physical correlations among features to identify attacks occurring over the system. For example, if an attacker performs a concealment attack on a subset of sensors in a system, this can break expected correlation between unmanipulated and manipulated sensors. Even a stateless detector, that only verifies the current state of the system could potentially detect such an anomaly. We can explain it with an intuitive example: consider a public place monitored by two CCTV devices. The attacker manipulates the images of one of the cameras (e.g., by replaying old images), but not the other one. Thus, an observer is able to detect a violation of spatial consistency as both cameras do not show the same scene. The same holds in an ICS, the same process is measured with multiple correlated sensors and an attacker replaying the values of few sensors causes inconsistencies.

7.4.3 Temporal Consistency

Temporal consistency refers to the temporal evolution of a sensor reading and how it unfolds according to the process physics and control action. Considering the state-space representation of a Linear Time-Invariant system (see Section 7.4.1), the update of the state x_{k+1} captures the temporal dependence between x_{k+1}, x_k and u_k according to A and B matrices. This relation can be used to predict the output of the system at time $k+1$, as the output depends on the estimated state at time k.

$$\hat{y}_{k+1} = C(Ax_k + Bu_k) + Du_k \tag{7.3}$$

Anomaly detectors should check the temporal evolution of the sensed value to verify if an anomalous unfolding is occurring over the system. For example, in a sensor spoofing attack, the temporal evolution of the spoofed data within might not follow the system's physics. Using the CCTV example used before, if only one camera is monitoring the public place, the start of a replay attack can be detected due to the sudden change of scenery (e.g., if the replayed video does not match the time of day). Thus, the attacker caused a temporal inconsistency in the video stream.

7.4.4 Statistical Properties

Sensor readings in the analyzed multivariate temporal series are characterized by a statistical distribution. Those properties are derived from the process that is generating

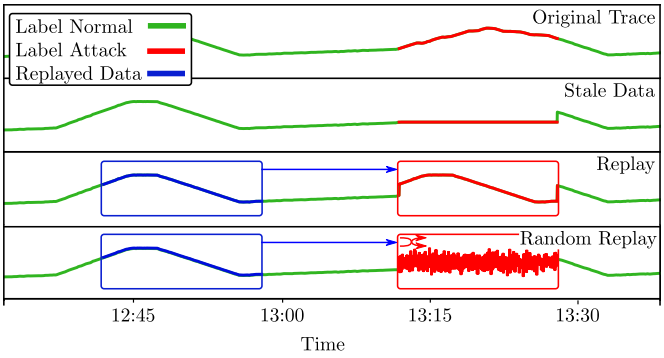


Figure 7.2: Visualization of attacks' effects on sensor data as seen by anomaly detector (simplified, single feature). 'Original Trace' contains anomalous data that triggers a detector.

the data, i.e., the matrices A, B, C, D of the system in State-space representation, control inputs and disturbances in the process and in the sensors v_k, w_k.

Anomaly detection can leverage statistical properties to spot anomalies. For example, each sensor reading is characterized by a proper mean and standard deviation, deviation from the expected distribution can trigger alarms, as the process generating the data has changed and this can be caused by the manipulation of the physical process caused by an attacker. In [S1], the authors showed that features in ICS datasets often have different statistical distributions in train and test data.

7.4.5 Verification of Consistencies

Given the three consistency properties, we verify if they are leveraged by anomaly detection systems. In order to do so, we identify three representative generic concealment attacks that break (one by one) the three consistency properties. Figure 7.2 provides a simplified graphical visualization of the three considered concealment attacks. If the attacks are successfully evading a detector, the detector is not correctly verifying the tested consistency property. As the attacks are designed to test the detectors, they are not necessarily optimized for unrelated metrics, e.g., minimal effort for the attacker. We note that those attacks are not present in the original datasets used for the evaluation of detectors.

Constrained Replay Attack. Constrained Replay is a variant of the full replay attack (e.g., discussed by the authors [134, P1]). Replay attacks conceal anomalies according to sensor readings observed in the past (e.g., by signal eavesdropping). This represents a relatively strong attack, as the attacker is required to record sensor readings for a certain amount of time before starting the attack. In constrained replay attacks, the attacker has a limited capability to spoof sensor readings and can replay sensor readings coming from a subset of PERA Level 2 area SCADA. Notably, also Stuxnet attack [202] resorted to a replay attack to conceal the true state of the system and avoid triggering alarms in the target industrial system.

While full replay attacks do not break any of the three consistency properties, constrained replay attacks are useful to understand if a detector correctly models the

Table 7.2: Attacks tested and their expected violation of consistency types. ●violates consistency.

	Consistency Tested		
Method	Spatial	Temporal	Statistical
Constrained Replay	●	-	-
Random Replay	-	●	-
Stale Data	-	-	●

spatial consistency among process sensor readings. When the attacker performs the replay on a subset of sensors/actuators the attack will break the correlations among features that hold in the system. Specifically, we apply the best-case scenario constraints proposed in Chapter 6.

Random Replay Attack. Random replay is a second variant of the full replay attack. It is the same as a full replay attack, except that the samples in the replayed data are temporally shuffled. E.g., assuming the attackers collected four multivariate samples $[y_1, y_2, y_3, y_4]$ where $y_k \in \mathbb{R}^s$, k is the time index, and s is the number of sensors in the network. After shuffling, the attacker replays the samples in the order $[y_3, y_1, y_4, y_2]$. It requires the same attacker capabilities of the aforementioned replay attack.

This attack is useful to understand if a detection scheme correctly models the temporal evolution of the sensor values i.e., if detectors correctly consider the data coming from a Markov Sequence or consider them Independent and Identically Distributed (i.i.d.).

Stale Data Attack. This attack implements a variant of the stale data attack discussed by Krotofil et al. [116]. In the stale data attack, a Denial-of-Service (DoS) attack is launched on the receiver of sensor data (e.g., the anomaly detector). The attack effectively prevents new sensor data from arriving. As demonstrated by Krotofil et al. [116], industrial end devices commonly handle such a loss of updates by assuming the last reported value is still current (e.g., to tolerate intermittent faults). As a result, the attacker can force a sensor reading to a specific value, by starting a DoS attack when that value is currently reported. The attack is unique in the sense that it represents a weak attacker in terms of required capabilities, as the attacker does not need to be able to eavesdrop on traffic or manipulate industrial protocols. The attacker only needs to perform a DoS attack, which is less effort to achieve.

This attack is useful to understand if detectors correctly model the statistical properties of sensor readings, when the attacker performs the DoS the observed statistical properties of the signal will change (e.g., variance becomes 0).

7.4.6 Mapping of Consistency to Attacks

Based on the three consistency types, we classify our three attacks in Table 7.2. The constrained replay attack tests spatial consistency because it changes a subset of the sensor readings without breaking the consistency with the non-spoofed sensors. The random replay attack tests the temporal consistency because the value of each sensor does not evolve according to the sensor process physics. Finally, the stale data attack

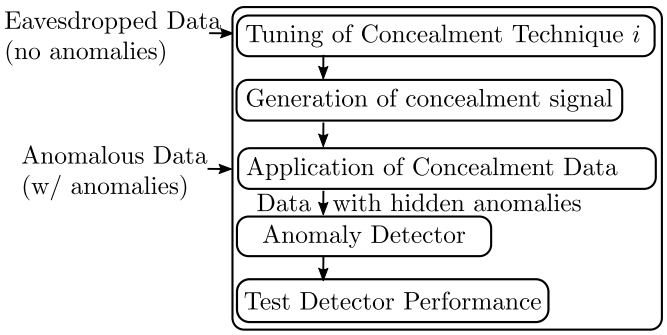


Figure 7.3: Overview of our framework for concealment attacks against an anomaly detector. The framework receives as input eavesdropped data (containing no anomalies) and anomalous data (containing anomalies). It then applies concealment techniques to the data. Concealment attacks are evaluated with the anomaly detector.

tests statistical consistency because it changes the statistical properties of the sensor signal producing a signal that has a different mean and no variance.

7.4.7 Abstract Framework Design

Given a target anomaly detector, our analysis requires eavesdropped data (with no anomalies) and anomalous data. We then generate concealment attack data based on the observed process features, and use this data to conceal the anomalous data. To perform the evaluation, we assess the performance of the detector over data containing anomalies. Then, we test the different concealment attacks applied to the anomalous data. If a concealment attack succeeds, the performance of the detector decreases, and the detector is observed vulnerable to that concealment technique. Concealment attacks are designed not only to attack detectors and reduce their Recall score but also they can be used to understand the properties of the physical process detectors fail to capture properly.

7.5 Evaluating the Anomaly Detectors

In this section, we explain how we used our framework design to test the six anomaly detectors and present the results. We apply to each of the six evaluated detectors the identified generic concealment attacks. We compare the results of the generic concealment with the detection results on the original (i.e., not enhanced with concealment attacks) dataset and with prior work Learning-based concealment attack [P1], where the attacker is assumed to manipulate sensor/actuators reading using a neural-network to conceal the anomalies on the system in real-time.

For two schemes (SFIG [62] and AR [85, 196, 9]), no reference implementation was available at the time of our experiments. Our re-implementations of detectors are publicly available.

7.5.1 Evaluation Methodology

In order to evaluate the performance of the anomaly detector, we observe how Accuracy Eq. 5.2, Precision Eq. 5.3, Recall Eq. 5.4, and False Positive Rate Eq. 5.6 scores change when the spoofing technique is applied to the data.

Given the original classification scores (e.g., when no spoofing is applied to data), concealment is effective if the Precision and Recall score reduces substantially. When those two scores reduce, towards 0, it means that the instances where the concealment was applied were misclassified moving them from being True Positives to False Negatives. Looking at the False Positive Rate (FPR) score we can also verify if the attacks are introducing False Positive in the Classification. If the FPR remains almost like the original, it means that the concealment did not induce any wrong classification (as expected since we are not spoofing data outside the boundaries of the attacks present in the dataset). Finally, since the datasets are unbalanced, with more samples of the negative class, the Accuracy score will not reach zero but at most the baseline where all the instances are labeled as the negative class.

7.5.2 Detector (Re-)Implementation

AR. We implemented the model-free AR detector as considered in prior work [85, 196, 9] with the CUSUM test. We implemented it in MATLAB and validated it on the Batadal dataset. The detector is based on an Auto-Regressive (AR) model trained over a univariate time series. The residuals of the AR model are used to compute a Cumulative Sum (CUSUM) statistic whose objective is to reveal a change in the process generating the data, i.e., spot anomalies in the system. We implemented this detector in MATLAB, using the System Identification Toolbox. We trained the model over Batadal data, we performed our experiments on sensor 'PRESSURE J302', i.e., the sensor that (alone) was allowing to detect the highest number of attacks (8 attacks over 14) with the considered detection method. We selected the AR model of order 20 with Best Fit criteria and tuned the CUSUM parameters using grid search and selected control_limit=5.5 and min_mean_shift_detect=1 obtaining as original detection performance Accuracy = 0.92, F1-score = 0.41, Precision = 0.79, Recall (TPR) = 0.28, FPR = 0.01.

PASAD. The implementation of PASAD anomaly detector [9] is available at `https://github.com/mikeliturbe/pasad`. PASAD analyzes every sensor univariate temporal series independently, for every sensor PASAD requires to be trained independently. We trained PASAD on the Batadal dataset over 'PRESSURE J302' sensor, we performed parameter tuning following the instructions provided in the original paper. Specifically, we used $N = 250$, $L = 250$, $r = 18$. The resulting original detection performance with this threshold over sensor J302 is Accuracy = 0.91, F1-score = 0.37, Precision = 0.74, Recall(TPR) = 0.24, FPR = 0.010.

SFIG. We re-implemented the anomaly detector based on the paper. We used Python 3 with the following libraries: Sklearn, Pandas, NumPy, SciPy. In this section, we summarize the parameters, and the assumptions we had to make to implement the detection system. ***Distribution Driven Strategy.*** We normalized the data between 0 and 1. We fitted Gaussian Mixture Models with at most 4 components for every sensor and took the one with the lowest BIC score. ***Event Driven Strategy.*** We set

Table 7.3: Performance of Autoencoders trained on Batadal.

Dataset	Acc.	F1	Prec.	Rec.	FPR
FC	0.950	0.729	0.864	0.631	0.012
LSTM	0.950	0.727	0.862	0.628	0.012
CNN	0.958	0.780	0.875	0.704	0.012

the threshold for the trigger $\epsilon = 0.05$, for Lasso we set $\alpha = 0.1$. ***Invariant Mining.*** Invariant mining is done with the CFP-growth++ algorithm. This algorithm is only available as open-source[65] in a Java library `http://www.philippe-fournier-viger.com/spmf/`. We used that library from our python script. Since the library is generating all the frequent itemsets that have the allowed minimum support, we parsed the output to identify the itemsets that do not break the non-redundant condition.

After re-implementing the detection mechanism, we were able to achieve a comparable result using the Batadal dataset. The resulting original detection performance is as follows. Accuracy = 0.93 F1-score= 0.58 , Precision = 0.75, Recall (TPR) = 0.47, FPR = 0.02.

AE. The implementation of the Autoencoder Based mechanisms using the three deep architectures (FC, LSTM, CNN) is available at the repository `https://github.com/scy-phy/ICS-Evasion-Attacks`. We leverage this implementation in this chapter. The input of the FC architecture is one set of sensor readings (i.e., 43 sensors), while for the LSTM and CNN the input is represented by the last two sampled set of sensor readings (i.e., 2x43 sensors). The performances of the three architectures are detailed in Table 7.3.

7.5.3 Attack Dataset Generation

We generate our attack data traces using a tool we wrote in Python 3, using the Pandas and NumPy libraries. The framework processes input training data (without anomalies) and test data (containing anomalies). Data should be organized in .csv format, where every row contains the sensor readings collected at a certain time step and every column represents a different sensor value. Test data is labeled, indicating whether the given row was anomalous or not. Optionally (in the constrained case) the framework takes the constraints on which variables can be spoofed. It then applies the presented concealment techniques to the data, and outputs the resulting augmented dataset (in .csv format). The framework can be applied to any similar dataset that meets the requirements.

Starting from the test data, the framework identifies the intervals in the dataset that are labeled as anomalous. The `evasion` function builds the dataset containing concealment attacks. It leaves the time intervals unchanged where ground truth reports 'normal' and applies the concealment attack to the time steps with ground truth 'anomalous'. The different concealment techniques are implemented as functions that apply the required spoofing to the given data. `Stale Data` attack replicates the sensor data as occurred in the last instance before the attack on the physical process started. `Replay` copies the data as found in the eavesdropped dataset. `Random Replay` copies

the data as found in the eavesdropped dataset and shuffles them temporally.

Specifically, we consider Batadal 'train dataset' and 'test dataset 1' for our evaluation. We generated 6 types of additional datasets (3 for the unconstrained attacks, and 3 for the constrained attacks). As we generate a dataset for each constraint value tested, we end up with 45 datasets for constrained attacks. Runtime for the attack dataset generation is less than two minutes for all attacks in total. As this dataset augmentation only has to be performed once for the evaluations, we find the runtime overall to be negligible. While we did not investigate real-time generation of concealment data (i.e., how an attacker might apply the concealment during an attack), we do not expect computational challenges.

7.5.4 Evaluation Results

We now provide the results of our experiments. Our evaluation is based on the analysis of the Recall score before and after the concealment attacks. A lower Recall indicates the attack is less likely to be detected. Note on False Positive Rate fluctuation in results: anomaly detectors classify instant t while aggregating all $t - n$ sensor readings before t. Datasets are composed of attacks interleaved by normal operations, if we spoof from instant a to instant b the classification outcome at time $b + 1$ depends on the manipulation that occurred between a, b, influencing the FPR.

7.5.4.1 AR

We attack the autoregressive (AR) detector with concealment attacks. An AR predictor is defined as 'good' if it generates residuals (i.e., prediction errors) distributed as white noise [26]. This anomaly detector uses the CUSUM algorithm to check whether the residuals are changing their distribution w.r.t. training phase. When an anomaly occurs, the CUSUM detects a change in the distribution of the residuals (i.e., no more distributed as white noise (WN): this is caused by the predictor that is not behaving optimally because of anomalous data). To succeed in concealment, the attacker has to modify the sensor signal such that obtained residuals do not surpass CUSUM control thresholds, i.e., residuals remain WN.

In Table 7.4 we report the results of concealment attacks. If we consider the Recall rate, we can notice that stale data attack reduces it to 0. The stale data approach is hiding the anomalies from the detector attack changing the value of the process with a constant, this makes the AR(20) model predicting the constant value that sends to 0 the CUSUM upper and lower statistics. In the case of the random replay attack, the spoofed signal causes sudden changes in the data, causing a change in the distribution of the residuals observed by CUSUM and triggering the alarms. This observation is consistent with [85], where it was observed that sudden changes in the process trigger alarms in AR detectors. In conclusion, the detector reliably models the temporal consistency of the signal but fails to model the spatial consistency (as it is univariate) and the statistical properties (as it does not detect the stale attack).

Comparing the results of the generic attacks w.r.t. learning-based attack [P1], we can observe that the neural-network based attack reduces the recall from 0.28 to 0.07 of

Table 7.4: Concealment attack results on AR model on Batadal sensor J302, unconstrained attack. †Note: technically NaN as the metric divides by 0.

Dataset	Rec.	Prec.	F1	Acc.	FPR
Original	0.28	0.79	0.41	0.91	0.01
Random Replay	0.29	0.88	0.44	0.92	0.01
Stale	**0.00**	0.00	$(0)^{\dagger}$	0.89	0.00
Learning-based [P1]	0.07	0.60	0.12	0.90	0.01

Table 7.5: Concealment attack results on PASAD Batadal Dataset sensor J302. Threshold = 635.1057.

Dataset	Rec.	Prec.	F1	Acc.	FPR
Original	0.243	0.741	0.366	0.910	0.010
Random Replay	**0.027**	0.530	0.051	0.894	0.003
Stale	0.471	0.779	0.587	0.929	0.016
Learning-based [P1]	0.241	0.740	0.364	0.910	0.010

the anomaly detector, but not as effective as the stale attack. Interestingly, an attack designed to target neural network based models, transfers to AR models.

7.5.4.2 PASAD

The detector treats the process data as a set of univariate time series, as in the case of the AR model this detector does not model spatial consistency. For this reason, we consider again the Batadal sensor J302, this allows direct comparison of detectors.

Table 7.5 summarizes the performance of PASAD when targeted with generic concealment attacks. Random Replay concealment techniques reduce the detector's performance. In two out of three proposed generic concealment attacks, we note that Recall decreases to 0.027 (from 0.243). In contrast, the detector performance increases for stale data attacks. The recall score decreases when we spoof the signal with a not physically plausible temporal evolution of the signal as in the random replay attack. This shows that the anomaly detector has learned the data distribution and not the physical process dynamics. This is also the reason why the stale data attack is detected. The stale data attack does not keep the statistical properties of the sensor signal, deviating from the expected statistical behavior. We can also note that if the stale data occurs on a value close to the process mean, the recall score decreases. Mathematically these results can be explained by analyzing Step 3 of PASAD anomaly detection scheme. PASAD projects training points in the signal subspace. Those projected points create a cluster in the projection subspace. Then, PASAD tracks the distance from the centroid of the cluster to identify anomalies. The centroid is defined as the sample mean of the lagged vectors. Our random replay attack fulfills the requirement of being projected within the cluster in the signal subspace. Despite its dynamics is not plausible, its departure score is lower than the threshold. At the same time, the stale data attack

Table 7.6: Concealment attacks results on SFIG detector on Batadal test dataset 1, unconstrained attack. †Note: technically NaN as the metric divides by 0.

Dataset	Rec.	Prec.	F1	Acc.	FPR
Original	0.47	0.75	0.58	0.93	0.02
Random Replay	0.15	0.49	0.23	0.89	0.02
Stale	**0.0**	0.00	(0)†	0.88	0.02
Learning-based [P1]	0.01	0.07	0.02	0.88	0.02

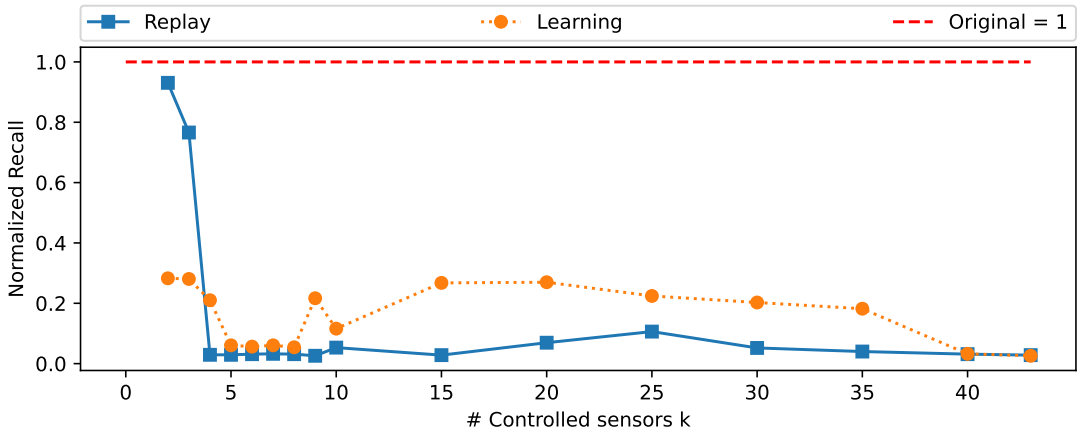


Figure 7.4: Impact of constrained concealment attacks on Recall score of the SFIG detector. 'Original' represents the Recall baseline of the model. Recall higher than 1 means that the detector is observing more anomalies after the concealment attack is in place.

surpasses the threshold because produces a different data distribution. In conclusion, the detector reliably models the statistical consistency of the signal but fails to model the spatial (as it is univariate) and temporal consistency.

Comparing the results of the generic attacks w.r.t. learning-based attack [P1], we can observe that the latter attack is not reducing the recall of the anomaly detector, so this attack does affect PASAD.

7.5.4.3 SFIG

We tested our attacks against SFIG anomaly detector trained on Batadal dataset. As this anomaly detector generates invariants aggregating all the sensors, we tested our framework in constrained and unconstrained settings.

Unconstrained setting. Table 7.6 reports the results of unconstrained attacks. Starting from the original detection Recall of 0.47, results show that the concealment attacks decrease dramatically the Recall score. When we apply the stale data attack to test the statistical consistency, Recall drops respectively to 0. This result indicates that this concealment attack was able to conceal the instances of anomalous data. If we consider what is going on in the anomaly detector, Recall close to 0 means that there are no

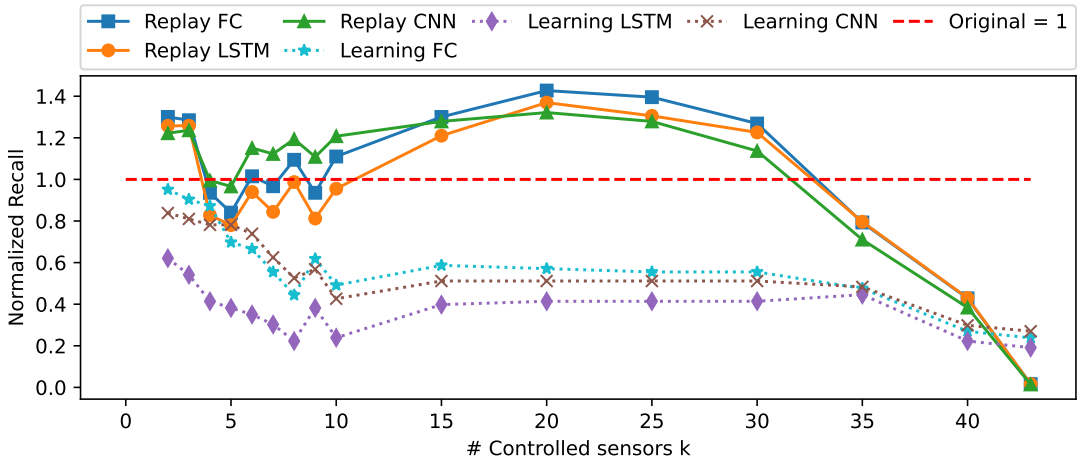


Figure 7.5: Impact of constrained concealment attacks on the autoencoder detectors. 'Original' represents the Recall baseline of the model. Recall higher than 1 means that the detector is observing more anomalies after the concealment attack is in place.

invariant rules violated by the attack. Indeed, Distribution Driven predicates are not violated since during training it often occurs that a sensor reading remains constant within two instants. At the same time, Event-Driven Predicates cannot be triggered. The spoofed signal reports that no events are occurring over the system. Hence, the system appears static, and the invariant-based detector stops checking invariant rules regardless of the data distribution of the samples. When we apply the random replay attack, the Recall score goes to 0.14, showing that the detector can detect (in part) the temporal inconsistencies.

Comparing the results of the generic attacks w.r.t. learning-based attack [P1], we can observe that the neural-network based attack reduces the recall of the anomaly detector, almost as effectively as the stale attack. Also in this case, it is interesting to notice that the neural-network based attack transfers to invariant based methods.

Constrained attack. In the constrained case where the attacker can only spoof a constrained set of sensor readings, i.e., they have compromised a subset of the PLCs and can spoof only certain sensors. For our experiments, we consider the Batadal constraints proposed by Erba et al. [P1]. As depicted in Figure 7.4, this detector fails to spot the constrained Replay. For example, when the attacker gains control of 4 out of 43 sensors (coming from at most 3 different PLCs/areas out of 9 in the network), the detection Recall drops to 0.0137. Comparing the results of the constrained replay attack w.r.t. constrained learning-based attack [P1], we can observe that also in this case, the neural-network based attack reduces the recall of the anomaly detector, but not as much as the constrained replay.

In conclusion, this detector fails to model spatial and statistical properties while it partially detects temporal inconsistencies.

Table 7.7: Performance of Autoencoders (Batadal).

Dataset	Rec.	Prec.	F1	Acc.	FPR
FC					
Original	0.631	0.864	0.729	0.950	0.012
Random Replay	**0**	0.000	(0)	0.883	0.012
Stale	**0**	0.000	(0)	0.883	0.012
Learning-based [P1]	0.151	0.606	0.242	0.899	0.012
LSTM					
Original	0.628	0.862	0.727	0.950	0.012
Random Replay	0.366	0.784	0.499	0.922	0.012
Stale	**0.003**	0.030	0.006	0.883	0.012
Learning-based [P1]	0.122	0.550	0.200	0.896	0.012
CNN					
Original	0.704	0.875	0.780	0.958	0.012
Random Replay	**0.004**	0.035	0.006	0.883	0.012
Stale	**0.003**	0.025	0.005	0.883	0.012
Learning-based [P1]	0.188	0.654	0.292	0.903	0.012

7.5.4.4 Deep Autoencoders

We tested the generic attacks against three different autoencoders. As in the previous case the detector considers the multivariate time series for detection and we perform unconstrained and constrained attacks. This builds upon and extends prior work experiments [P1] to relate to the consistencies proposed in this chapter, which were not considered before.

Unconstrained setting. Table 7.7 reports the results of unconstrained attacks applied to the three autoencoder architectures. Starting from the original detection Recalls (respectively 0.631 for the FC, 0.628 for the LSTM and 0.704 for the CNN), results show that the concealment attacks are capable of evading the detectors. When we apply the stale data attack to test the statistical consistency, Recall drops close to 0 for all the three architectures. These results indicate that this concealment attack was able to conceal the instances of anomalous data and the detectors are not correctly exploiting the statistical properties of the signal. When we apply the random replay attack, the Recall score goes to 0 for the FC and CNN architectures, showing that those detectors are not detecting temporal inconsistencies. On the other end, the recall of the LSTM architecture targeted with the random replay is 0.366, this shows that correlating the last two sets of sensor readings in the input layer allows detecting anomalous temporal evolution of the process.

Comparing the results of the generic attacks w.r.t. learning-based attack, we can observe that the learning-based attack reduces the recall of the anomaly detector, but not as well as the generic concealment attacks. The only exception is the random replay for the LSTM, surpassed by the learning-based attack.

Constrained attack. As in the previous section, the second experiment studies the

Table 7.8: Summary of findings. Vulnerability to the generic concealment attacks. ✓detected (not vulnerable), ✗non detected (vulnerable), N.A. not applicable as the detector considers the univariate time series.

	Detected		
Detector	Constr. Replay	Random Replay	Stale
AR [85, 196, 9]	N.A.	✓	✗
PASAD [9]	N.A.	✗	✓
SFIG [62]	✗	✓	✗
AE FC [188]	✓	✗	✗
AE LSTM [77]	✓	✓	✗
AE CNN [112]	✓	✗	✗

constrained case where the attacker can only spoof a constrained set of sensor readings i.e., they have compromised a subset of the PLCs and can spoof only certain sensors. We use the same constraints also for this model. As depicted in Figure 7.5, the detectors identify the spatial inconsistencies introduced by the replay attack, also when the attacker controls almost all the sensor readings (i.e., 40 out of 43) the detection recall is around 40% of the original detection score. In conclusion, all the three autoencoders model properly spatial consistency, while they fail to model statistical properties. FC and CNN also failed to capture temporal properties of the system in contrast to LSTM. Comparing the results of the generic attacks and learning-based attack [P1], we observe that the neural-network based attack reduces recall of the anomaly detector almost as effectively as the stale attack.

7.5.5 Summary of Findings

With respect to **RQ7.2**, results show a varied performance of detectors w.r.t. the three considered attacks and the related consistency properties, none of the detectors is resilient to all the three considered attacks but at most to two (i.e., AE LSTM resilient to constrained replay and random replay). Those attacks break the physical properties of the system and are, in theory, easy to spot. Model-free detectors fail to exhaustively abstract Spatial, Temporal, and Statistical consistencies to perform anomaly detection.

With respect to **RQ7.3**, results show that neural-network based attacks [P1], can effectively be used to conceal the effects of attacks on the physical process for AR, SFIG, and Autoencoders, but they fail to succeed against PASAD detector. On the other hand, the concealment performance (reduction of the recall score), is always in favor of one of the generic attacks. Moreover, learning-based attacks from prior work require real-time computation to adapt the spoofing pattern to the current sensor readings, while generic concealment attacks do not need to be adapted in real-time.

Countermeasure. Given the results for RQ7.2 and RQ7.3, accurate attack detection without detailed a priori process models remains an open issue. In the next section, we leverage the consistency properties to construct and test a data-driven model-based detector that reliably detects process anomalies and concealment attacks.

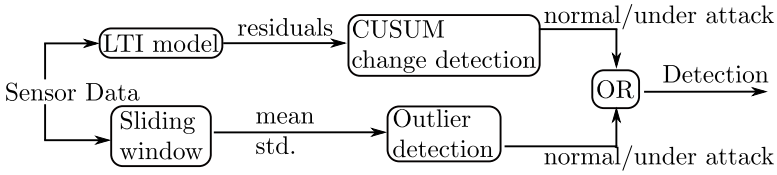


Figure 7.6: Overview of the ensemble of detectors, LTI model and sliding window statistics are used to build a detector resilient to concealment attacks.

7.6 Data Driven Model-based Detector

Given our results in the Section 7.5, accurate attack detection without detailed a priori process models remains an open research question. In the following, we answer **RQ7.4** by constructing and testing a data-driven model-based anomaly detector that leverages the physical properties of the process and reliably detects both the process anomalies in the data and concealment attacks. Like our assessment framework, the proposed ensemble detector code is publicly available. Following our attacker model and capabilities (Section 7.3), we consider white-box attackers (i.e., with process knowledge and detector knowledge) out of scope in our evaluation. As shown in prior work [195, P1], that in white-box setting attacks cannot be fully-prevented can be at most delayed or reduce their impact. Moreover getting this white-box process/detection knowledge is challenging (if not impossible) [82, 166, 106].

Detector Architecture. The detector is based on an ensemble of two complementary detectors trained on process data collected on the system in normal operation condition (i.e., without anomalies). The detectors in the ensemble are designed to detect process anomalies and concealment attacks. Figure 7.6 gives an overview of the ensemble architecture. The first model in the ensemble is using an identified Linear Time Invariant (LTI) model of the process to predict future system behavior, which enables us to compute residuals which we use for a stateful (CUSUM) detector. This part of the ensemble allows us to identify spatial and temporal inconsistencies among features. The second model is a sliding-window based statistical outlier detector, required to detect remaining attacks that manipulate the signal without inducing physical inconsistencies (e.g., Stale Data attacks). The predictions are combined using the OR operator. The anomaly detector benefits from the ensemble as each component is trained to abstract certain properties of the physical process (see Figure 7.7).

No A-Priori Process Model. In contrast to other model-based schemes (e.g. [44, 158]), our data-driven model-based detector does not require explicit a priori process models or templates to be constructed and trained. Instead, our two models leverage Subspace-based State Space System Identification techniques and statistical analysis, respectively.

7.6.1 Data-driven Stateful LTI Detector

Formulating the precise system equation for a complex CPS is challenging, requires in-depth process knowledge, and might lead to deal with complex non-linear equations. This part of the ensemble uses an LDS-model based stateful detector (see [195]), which

Algorithm 2: Rolling window detector training

```
function RollingDetector(trainData, window)
    features ← []
    percentiles ← []
    for sensor in listOfSensors do
        hMatrix ← hankel(trainingData[sensor], window)
        // mean of rows
        slidingMean ← mean(hMatrix)
        // std of rows
        slidingStd ← std(hMatrix)
        // indexes of windows where variance is zero
        indexesZeroStd ← slidingMean[slidingStd==0]
        // find the mean of the sliding windows means with
           zero variance
        meanZeroStd ← mean(indexesZeroStd)
        // check if the mean of the windows with zero
           variance is zero, or empty (i.e., there is no
           window with zero variance, in all the windows the
           sensor has been updated at least once)
        if meanZeroStd in [0,NaN] then
            // find minimum non-null percentile of the
               slidingStd vector
            percentile ← findPercentile(slidingStd)
            features.append(sensor)
            percentiles.append(percentile)
    return features, percentiles
```

effectively requires a system characterization in form of a set of LTI equations. While such designs were discussed in prior work [195], no concrete implementations have been released, there was no evaluation w.r.t. the attacks of the datasets in this chapter, or our newly contributed concealment attacks for those datasets.

In order to not rely on a-priori process characterizations, we derive an approximation of the LTI representation leveraging the Subspace-based State Space System Identification techniques (n4sid algorithm [197]). Through this technique, we approximate the coefficients of the system model (i.e., matrices A, B, C, D and disturbances K) without explicit knowledge of the system equations. Specifically, we consider the water tank levels as output values of the system, while all the other continuous sensor readings as input data.

Once the model is identified, a classification function of the one-step-ahead prediction residuals (CUSUM, SVM, etc.), can be used to identify attacks to the spatial and temporal properties. Specifically, for each output of the LTI model, we use the CUSUM algorithm with change detection as a classification function for residuals.

Algorithm 3: Rolling window detector testing

```
1  function anomalyDetection(features, percentiles, window,
   testData)
2      predictions ← []
3      for sensor in features do
4          hMatrix ← hankel(testData[sensor], window)
           // mean of rows
5          slidingMean ← mean(hMatrix)
           // std of rows
6          slidingStd ← std(hMatrix) // for each window verify
              if the variance is lower than the lowest non-null
              percentile observed during training
7          for i in len(slidingMean) do
8              if slidingMean[i] > 0 and slidingStd[i] <
                percentiles[sensor] then
9                  predictions[i] ← 1
10             else
11                 predictions[i] ← 0
12     return predictions
```

7.6.2 Statistical Outlier Detection

In order to detect the attacks that violate the statistical properties of the system, we propose a sliding window-based outlier detection method that identifies changes in sensor statistics. Algorithm 2 shows how the detector is trained, while Algorithm 3 shows how the detector is used to perform anomaly detection. The intuition is to detect changes in the process variance (e.g., the changes introduced by the stale data attack). Using l samples of training data, we apply a sliding window of length w to each sensor reading in the dataset, obtaining $k = l - w$ traces of data (per sensor). For each trace in k we compute its mean and standard deviation. For sensors that have a variance greater than 0 (or variance 0 when the mean value is 0) in all the k traces (i.e., the sensor updates its value at least once in w timesteps), we use an approximate binary search algorithm to find the minimum non-null percentile of the sliding window standard deviation distribution and use it as a threshold. At test time, we compute the sliding window statistics for the sensors that satisfy the conditions in the training set. If the variance of a window is greater than 0 but lower than the sensor's threshold an alarm is raised.

7.6.3 Classifier Contributions in the Ensemble

Figure 7.7 shows an example on how the two detectors in the ensemble complement each other. The attacker launches a Stale Data attack (i.e, the attack breaks statistical properties of the sensor readings). We note most of the attacks are detected by both

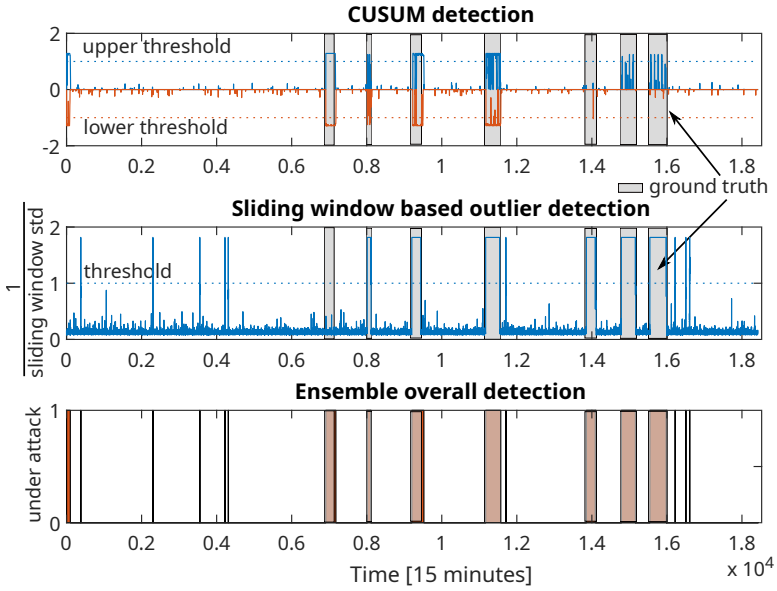


Figure 7.7: Contribution of classifiers in the ensemble. Detection of constrained stale data attack (normalized thresholds for readability). Some of the attacks are detected exclusively by one of the two detectors in the ensemble. The OR condition between the predicted labels of the two models allows to detect attacks with high precision and recall.

the models in the ensemble, but others are detected by either one of the two as the spoofed signal might trigger exclusively spatial/temporal or statistical inconsistencies.

7.6.4 Results

We train the predictions of the two methods presented in 7.6.1, 7.6.2 tested on the Batadal dataset. The LTI system was modeled using the 7 water level sensors as output data while all the other 24 continuous sensor readings in the Batadal dataset were used as input. We identified the model of order 11 using n4sid MATLAB implementation. The residuals of the model are classified using CUSUM. The Statistical based detector was trained using all the 31 continuous sensor readings. Table 7.9 reports the results of the ensemble of methods on the Batadal dataset attacked with the unconstrained concealment attacks. Stale and random replay attacks are detected with a high recall rate. We can observe how the two components of the ensemble contribute to the detection, for example in the random replay attack the temporal inconsistency is triggering the LTI detector but not the Sliding window detector (as expected since there is no statistical change in the data). Figure 7.8 reports the results of the constrained replay attack in Batadal data (same constraints used in Figure 7.5), also constrained replay is detected by our countermeasure. The main contribution to the detection, in this case, is given by the LTI model. The performance of the ensemble in this scenario is comparable to the autoencoder models (Figure 7.5), although the recall of the ensemble model decreases faster when the attacker controls 30 sensors or more.

Table 7.9: Contribution of classifiers in the Ensemble tested on Bataldal dataset. For each experiment, the table reports the performance of each classifier in the ensemble (LTI, Sliding) and their overall performance (Ensemble)

Dataset	Acc.	F1	Prec.	Rec.	FPR
Original Attacks					
LTI	0.91	0.29	0.74	0.18	0.01
Sliding	0.94	0.62	0.93	0.46	0.00
Ensemble	0.95	0.71	0.86	0.61	0.01
Stale					
LTI	0.98	0.91	0.91	0.91	0.01
Sliding	0.99	0.97	0.96	0.99	0.00
Ensemble	0.99	0.94	0.89	1.00	0.01
Random Replay					
LTI	0.99	0.96	0.93	1.00	0.01
Sliding	0.89	0.00	0.04	0.00	0.00
Ensemble	0.99	0.95	0.91	1.00	0.01
Learning-based Attack [P1]					
LTI	0.97	0.88	0.81	0.96	0.03
Sliding	0.89	0.01	0.12	0.00	0.00
Ensemble	0.97	0.87	0.79	0.96	0.03

7.6.5 Summary of Findings

We built an ensemble detector that is constructed without leveraging process knowledge and outperforms prior model-free detectors. Our proposed ensemble can detect both the original process anomalies contained in the datasets and the concealment attacks considered in the work (i.e., both generic concealment attacks and Learning-based [P1]). Our model outperforms prior work model-free approaches as it is capable of detecting spatial, temporal, and statistical inconsistencies in the data.

7.7 Conclusions about Generic Concealment Attacks

In this chapter, we introduced the concepts of spatial, temporal, and statistical consistency for process-based anomaly detectors (RQ7.1). To assess which detectors verify which consistency, we leverage three generic concealment attacks. We then designed and implemented a framework to add those attacks to common datasets, and evaluated six model-free detectors (RQ7.2). Our evaluation results show that the considered attacks are effectively evading prior work detectors, which demonstrates that detectors are not verifying all three consistencies. Although our attacks were designed to test consistencies (and were not particularly optimized for performance), we noted that they were surprisingly effective even compared to more optimized prior work. Our

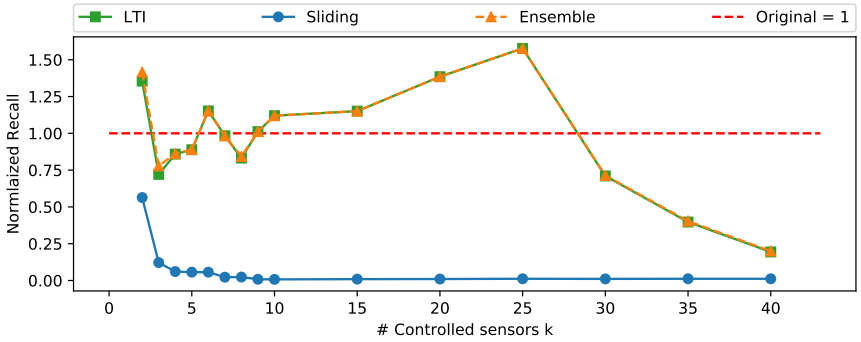


Figure 7.8: Constrained replay attacks results on our countermeasure. This plot shows the impact of constrained concealment attacks over the Recall score. The plot shows the contribution of the classifies in the ensemble

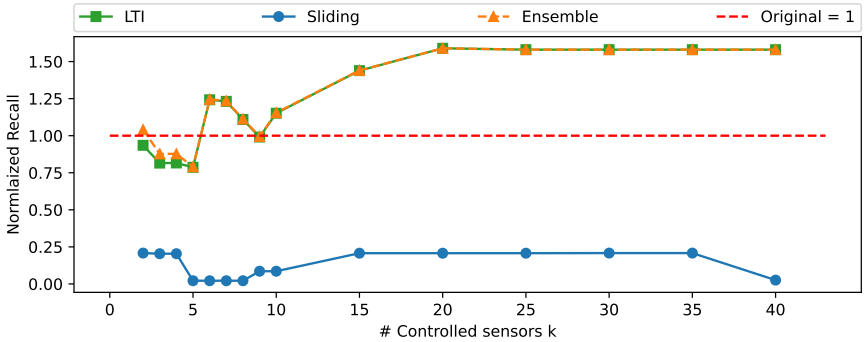


Figure 7.9: Constrained Learning-based (P1) attacks results on our countermeasure. This plot shows the impact of constrained concealment attacks over the Recall score. The plot shows the contribution of the classifies in the ensemble

attacks reduced the Recall of AR models from prior work from 0.28 to 0.0, of PASAD [9] from 0.24 to 0.02, of SFIG [62] from 0.47 to 0.0 of FC autoencoder 0.631 to 0, of the LSTM autoencoder from 0.628 to 0.003 and of the CNN autoencoder from 0.704 to 0.003. The weaknesses we demonstrated in the anomaly detectors show that (despite the good detection performance of the original schemes), the detectors are not able to detect adversarially manipulated physical system properties. Our results also show that generic concealment attacks are possible, in contrast to prior work that assumed to have a white-box knowledge of the target system [195]. The analysis and the results in our contribution highlight the need for more complete datasets and critical analysis of model-free detectors to evaluate their performance. As such, we see our contribution to the discussion about the resiliency of anomaly detectors when analyzed against targeted manipulations.

We have compared the identified generic concealment attacks with prior work learning-based attacks (RQ7.3). The identified generic attacks are also better than prior work although they do not require real-time adaptation of the sensor readings. Identified generic attacks focus on one consistency each, and they perform better than

prior work attacks. On the other hand, we show that learning-based attacks transfer to other models.

Finally, we leverage the identified physical properties of the process to build an ensemble detector capable of detecting process anomalies and concealment attacks (RQ7.4).

8

White-Box Concealment Attacks

White-Box Concealment Attacks Against Anomaly Detectors for Cyber-Physical Systems

8.1 Introduction

In Chapter 7 ([P3]) we demonstrated that generic black-box concealment attacks on general anomaly detectors are possible, those attacks are effective but are not optimized and manipulate a large number of sensors, over many samples. It is unclear how optimal those attacks are—we need a baseline to compare against. White-box concealment attacks by a less constrained, more knowledgeable attacker could provide such a baseline, but those attacks were only investigated for the specific subclass of Deep learning-based anomaly detectors in Chapter 6. Thus, the threat posed by white-box concealment attacks on general anomaly detectors is unclear, and in particular, the minimal perturbation required to achieve misclassification (by strong attackers) is unknown for each detector.

In this chapter, we bridge this research gap by addressing three research questions: **RQ8.1** How resilient are anomaly detectors for cyber-physical systems against white-box concealment attacks? **RQ8.2** Can white-box attacks efficiently compute manipulations at runtime? **RQ8.3** How do the white-box attacks perform compared to prior work black-box attacks?

To address the aforementioned research questions we tackle two research challenges: **C8.1** The attacker manipulates dynamic streaming data, i.e., the attacker cannot retroactively change past values, or predict future process sensor values. **C8.2** General detectors are not guaranteed to optimize a differentiable loss function for detection (in contrast to Deep Learning-based detectors). We address C8.1 by implementing and evaluating a method that manipulates only the current sensors' observations and show that it is still possible to minimize the detection function loss. We address C8.2 by proposing a method to re-write non-differentiable classification functions as differentiable and hence allow concealment attacks.

List of Contributions. The main contributions of the chapter are:

- Designing an effective general purpose white-box concealment framework for anomaly detection systems.
- Formulation of loss-free detectors (i.e. process invariants), as loss-based.
- Evaluation of proposed white-box attacks with real testbed data against five state-of-the-art anomaly detection systems.
- Comparison of the proposed white-box concealment with prior attacks.

8.2 CPS: Background and Related Work

Anomaly detection for CPS. A number of process-based anomaly detection techniques were proposed in the literature. They leverage the characteristics of the physical process to detect deviations in the process data caused by attacks [31]. *i) Residual-based approaches* are trained to minimize a loss function (usually Mean Squared Error), between the expected and observed sensor readings. To detect anomalies, the loss between input and output is monitored, if it exceeds a threshold an alarm is raised. In this

Table 8.1: Summary of anomaly detection families proposed in prior work in the context of CPS. The table reports the approach used for detection, and the detectors that we analyze in our evaluation (○= no, ●=yes). We skip DNN as it was analyzed before (see Chapter 6.

	[85]	[9]	[195, 44]	[7, 41]	[77, 112, 188]	[5, 62]
Approach type	AR	SVD	LTI	SVM	DNN	Invariants
Classification Differentiability	D	D	D	D	D	N
Prior WBC analysis	○	○	○	○	●	○
Analyzed in this chapter	●	●	●	●	○	●

category, we find control theoretic approaches e.g., Auto Regressive (AR) models [85] and Linear Time Invariant (LTI) models [195, 44], and machine learning approaches e.g., Support Vector Machines [7, 41] and Deep Neural Networks [77, 188, 112]. *ii) Invariant-based approaches* consist of rules that describe conditions that always hold in a given state on the CPS [5]. Those rules are often written based on detailed process knowledge [5, 62].

Evasion attacks against CPS. In Adversarial Machine Learning, evasion attack refers to the setting in which an attacker modifies a sample to induce misclassification in a classifier [22]. In the context of Advanced Driver-Assistance System (ADAS) several attacks were proposed e.g., against LIDAR [30], location estimation [175]. In the context of CPS anomaly detection, white-box attacks against Deep Learning models [214, P1] were demonstrated. Also, generic black-box evasion techniques were proposed [P3]. Table 8.1 summarizes prior work in the field of anomaly detection for CPS, and reports which models were analyzed before for white-box concealment attacks. In this chapter, we focus on models proposed in prior work but not analyzed so far against white-box concealment.

8.3 System and Attacker Model

Consistent with the system and attacker model introduced in Chapter 5, we assume a cyber-physical system equipped with an anomaly detection system. The attacker perturbs the physical process to cause an anomaly, at the same time, a concealment attack is launched to hide an ongoing process anomaly. In this chapter, we assume an Unconstrained White box attacker $(\mathcal{D}, \mathcal{X}, f, w)$ (see Chapter 5.4.3).

We measure the cost of the attack with respect to the number of features that are manipulated using the L0 norm (independent of the modification amount, i.e. L2 norm), as the effort is in compromising the communication channel, and at that point, arbitrary values can be set [P1]. In practice, we allow any perturbations within the operational limits of the respective sensor or actuator [156].

8.3.1 Research Goals and Challenges

We address the three open research questions presented in the introduction. While addressing the three research questions we tackle the following research challenges:

C8.1 The attacker manipulates dynamic streaming data on the fly, which means that the attacker i) iteratively manipulates each value sequentially without knowing future values in advance; ii) adapts the strategy according to previous values stored in data logs without altering them. This is imposed by the Cyber Physical Systems, where the attacker is assumed to perform sensor spoofing exploiting communication channels vulnerabilities. **C8.2** Not all general detectors are guaranteed to have differentiable loss functions (in contrast to Deep Learning based detectors). Thus, we need a general technique to attack different detectors even in absence of a loss function. For example, the detector [62] represents the current sample as a boolean vector (each element representing whether a specific invariant was violated). For this reason, we cannot use gradient-based methods (for example) to find optimal evasion samples.

Our main goal is to assess whether additional knowledge on detection mechanisms (i.e. white-box attacks) allows the attacker to perform better compared to black/grey-box attacks discussed in prior work [P1, P3]. This allows us to assess the robustness of CPS anomaly detectors, i.e., the minimal number of communication channels (features) that need to be controlled by the attacker to avoid detection.

8.4 Proposed approach

We design a generic white-box concealment attack for CPS anomaly detectors. In this section, we start proposing the general framework that can be applied to attack prior work anomaly detectors.

8.4.1 White-box Concealment Attacks (WBC)

We translate the white-box concealment attack (WBC) objective (Equation 5.12) into an error minimization problem (Equation 8.1)

$$\begin{aligned} \text{minimize} \quad & Loss_{x_{adv}}(x_{adv}, tc) \\ \text{where} \quad & tc = \text{target class} \end{aligned} \tag{8.1}$$

Then, we induce targeted misclassification (to achieve the goal in Equation 5.12) inspired by the Fast Gradient Signal Method (FGSM) [80] proposed originally for the domain of image manipulation.

$$\delta = -\epsilon * \text{sign}(\nabla_x Loss(x, tc)) \tag{8.2}$$

Every anomaly detection method has a different classification function, and consequently a different loss (if explicitly present), for this reason, this generic method is suitable to be applied to different categories of anomaly detectors.

The perturbation in Equation 8.2 is iteratively applied until the concealment attack is successful and the detector no longer flags the anomaly (Equation 5.12). Two attackers can be considered in this setting (we will compare them in Section 8.6). The first continue iterating until the objective (Equation 8.1) is minimized, and the second continues until the classification label is changed, but the objective is not necessarily minimized.

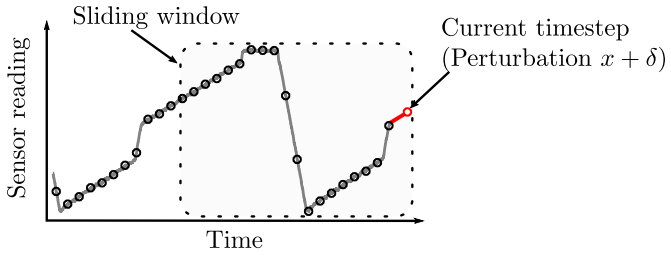


Figure 8.1: Challenge **C8.1**. For each time slot, the attacker can only manipulate the latest sensor reading without knowing future values. We note the attacker cannot retroactively modify previous (manipulated or original) values. Eventually, the data considered in the sliding window will exclusively process values that were manipulated before.

8.4.2 Attacking Detectors with Differentiable Classifiers

We now address the research challenge **C8.1**: on-the-fly manipulation of streaming data (see Figure 8.1). Residual-based anomaly detectors classify anomalies based on the residual error between the sensors and actuators readings x and a predicted output value o from the anomaly detection classifier. That classification is performed over a sliding window of past observed values (i.e., $[x_{t-n}, \ldots, x_t]$).

In our scenario, the attacker can not simultaneously manipulate each value in this sliding window (as it would require post-hoc change of data), only the current sensor reading x_t can be manipulated. This introduces a novel constraint on the attack as the attacker has to minimize the residual loss acting on the last observed sample, and cannot globally minimize the loss function. We account for this additional constraint in our evaluation.

For example, to perform the WBC attack for residual-based detectors, we model the residuals by using the Mean Squared Error loss (Equation 8.3). Then, we compute the partial derivative of the mean squared error w.r.t. x (Equation 8.4) and apply directly FGSM to it (Equation 8.2).

$$Loss(x, o) = \frac{1}{2}(x - o)^2 \qquad (8.3) \qquad\qquad \nabla_x Loss(x, o) = x - o \qquad (8.4)$$

8.4.3 Attacking Detectors with Non-Differentiable Classifiers

Invariant-based anomaly detectors [5, 62] classify anomalies based on the coherence of the system sensors and actuators w.r.t. a set of process invariant rules. When invariants are used, detectors check if some invariant rules are not fulfilled and raise an alarm consequently.

$$\text{Given an invariant rule } R\text{:} \quad A \rightarrow B \quad \text{(read as: if A then B)} \qquad (8.5)$$

Where A is the antecedent and B is the consequent of the invariant rule. Antecedent and consequent of a rule, consist of set of predicates over certain sensors and actuators (e.g., valve_status $= 1$ and sensor_value < 4). An anomaly is identified if predicates in the antecedent A are all satisfied but not all predicates in consequent B are satisfied.

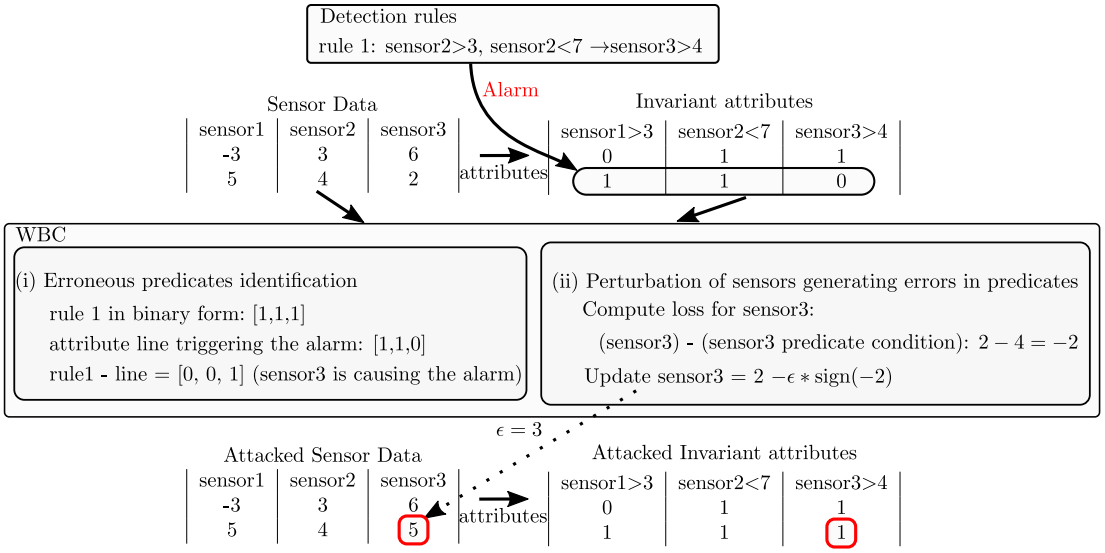


Figure 8.2: Challenge **C8.2**. WBC concealment against detectors without loss function. Invariant-based detection works by checking whether the sensor readings satisfy the invariant rules, without relying on predictive models (no loss function). To apply our WBC attack, we re-formulate invariant-based methods as loss-based. We manipulate the sensors and actuators readings according to the difference (loss) between the desired state specified by the rules and the current state.

This method does not employ a loss function. In order to evade such detectors we need to consider the research challenge **C8.2**, i.e., we need to formulate the invariant-based approach as a loss-based method. Specifically, to evade the detector an attacker is required to modify the sensor readings in such a way that the predicates in B are fulfilled[1]. In order to do so, we decompose the attack in two steps (Figure 8.2 provides a toy example of the method).

(i) Erroneous predicates identification. In the first step we identify which predicates trigger the anomaly in B. To do so we perform the set difference between the predicates in the rule R and the predicates observed in the system P (Eq. 8.6).

$$R \setminus P \tag{8.6}$$

Practically predicates are represented by Boolean conditions (i.e., boolean vectors where the position represents a certain invariant and the value 1 or 0 represents if the invariant condition holds). We identify the predicate that does not match the triggered rule performing the difference of such vectors.

(ii) Perturbation of sensors generating errors in predicates. In the second step, for the predicates that are erroneous we need to perturb the data related to that predicate to induce the change in the generated predicates. To guide sensor reading perturbation we can consider the desired value (i.e., the condition required by the predicate) of the erroneous predicates as our target value. This step can be performed

[1]Alternatively the attacker can deactivate a rule by violating one condition in A, but this does not give guarantees about other rules that might be triggered by the modification.

by substituting the desired value directly in the sensor reading if the predicate is a direct equality or inequality over the sensor value (e.g., sensor = 3). Otherwise, if the predicate aggregates more information about a sensor reading (e.g., Gaussian Mixture Models over sensor value updates), we formulate the problem as a Mean Squared Error minimization as in Section 8.4.2, and compute the perturbation using Equation 8.2 and using the loss as in Equation 8.3.

8.5 Implementation and Evaluation Setup

In this section, we provide details about the implementation setup, the target anomaly detection systems, and the dataset used for evaluation. Based on the categories of detectors identified in Section 8.2 and the analysis of prior work white-box concealment attacks in Table 8.1 we selected the target detectors according to three main criteria: (i) diversity of the detection technique (ii) not covered by prior work studies on white-box concealment attacks (iii) code availability for the detector. Our selection covers the research gap in the field of white-box concealment attacks on CPS anomaly detection. We consider five different anomaly detectors proposed in relevant prior peer-reviewed publications; namely, Auto Regressive model [85], Linear Time Invariants [195], Support Vector Machines [41] [9], Process Invariants [62] (see Table 8.1), and for each, we apply our proposed approach to achieve misclassification.

8.5.1 Attack Implementation and Hardware Setup

All experiments were performed on a laptop, equipped with Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz, and 16GB of RAM. Experiments were performed either using Matlab 2019a, or Python 3.8.10 (depending on detector sources).

Implementation of the attack required: 201 lines of Matlab code for the AR model [85], 249 lines of Matlab code for the LTI model [195], 287 lines of python code for the SVM [41] in this case we relied on the secml [131] library for gradients calculation by creating a wrapper for sklearn OneClassSVM, 324 lines of code for the PASAD detector [9], and 490 lines of code for the SFIG detector [62]. The code of our attacks is available at `https://github.com/scy-phy/whiteboxDimva23`.

8.5.2 Linear Time Invariant Models

We identified an order 4 LTI model for the SWaT dataset. We use 22 sensors as input of the system, and the 3 tank level sensors as the output of the model. ***Availability.*** we relied on the re-implementation by Erba et al. [P3] and we adapted it to work with SWaT dataset.

8.5.3 SVM

Following the guideline in the paper we performed a grid search to tune the parameters of the SVM. The resulting model is OneClassSVM with linear kernel, γ=0.01, ν=0.02. The SVM is trained on the sensor readings (π,π') measured at d steps from each other, we fine-tune the parameter d. With our experiments, we tested d = 1, 10, 100, and

1000 seconds and found the best performance at 1 second. We note that the simulator used in the original paper has a faster sampling rate (5 milliseconds) than the actual SWaT testbed sampling rate (1 second).

8.6 Evaluation Results

In this section, we present the results of our evaluation. To answer to **RQ8.1**, we applied the five aforementioned detection mechanisms to the SWaT dataset [129] and attacked them with the proposed WBC. To answer **RQ8.2**, we verify the computational runtime of the proposed approach and the cost of the perturbations. Finally, to answer to **RQ8.3**, the results of the WBC attack methodology are compared against the performance when no concealment was applied to the data, and against the black-box attacks for CPS detectors [P3].

For our proposed WBC attack we consider three variants. Namely, *WBC baseline*, where the WBC attack is applied to every set of sensor readings labeled as 'anomalous' as ground truth (i.e., the attacker is manipulating the physical process), regardless if they are detected as anomalous or not. In this setting, the attacker iterates until the objective (equation 8.1) is minimized. This is the same setting considered by the attacks proposed by Erba et al. [P3], and we use it for comparison. *No True Positives (WBC NTP)*, in this setting the WBC is applied to every set of sensor readings labeled as 'anomalous', which is also detected as anomalous by the anomaly detection system (i.e., physical anomaly correctly detected by the anomaly detection system). Finally, we consider the *No Alarms (WBC NA)*, in this setting the WBC is applied to every set of sensor readings that are detected as anomalous by the anomaly detection system (i.e., conceal also false positives). In WBC NTP and WBC NA settings, the attacker iterates until the label is changed.

We note that since there is the white-box assumption on the target detector, the attacker is assumed to access the prediction of the detector. Moreover, since the physical process manipulations are under the control of the attacker, the attacker knows when the physical process anomaly is occurring on the system (i.e., 'anomalous' ground truth in the SWaT dataset).

Evaluation Metrics. To assess the impact of the attack over the detection capability of the classifier we consider the following metrics: Accuracy, F1 score, Precision, Recall, and False Positive Rate. In particular, the Recall score gives us information on how the attack is capable of concealing the true state of the system from the anomaly detector. Elapsed time is measured to assess the mean computational overhead required by the WBC attack. Specifically, we measure average the time required to compute an adversarial example. Finally, we measure the Euclidean distance (L2) between the original sample p and the perturbed sample q to assess the perturbation required on the features by the attack. Moreover, to evaluate the minimal number of features under the control of the attacker we compute the Hamming distance (L0), as the number of sensors/actuators that were changed by the attack.

Table 8.2: WBC Attack on the AR model trained over SWaT sensor LIT301 (used as reference in prior work (9)). The WBC attacks evade the anomaly detection system (see original recall vs. WBC recall). μ indicates the mean, and σ the standard deviation. †Note: technically NaN as the metric divides by 0.

						Elapsed(ms)		Euclidean D.		
Data	Acc	F1	Prec	Rec	FPR	μ	σ	μ	σ	N
Original	0.797	0.254	0.227	0.288	0.134	-	-	-	-	-
Prior Work [P3]										
Replay	0.775	0.088	0.086	0.091	0.131	-	-	13.592	48.322	541
Random R.	0.832	0.501	0.389	0.702	0.151	-	-	12.832	45.903	541
Stale	0.788	0.186	0.173	0.201	0.131	-	-	17.341	55.804	522
Our WBC										
baseline	0.858	(0)†	0.000	0.000	0.024	0.004	0.014	11.046	40.578	515
NTP	0.860	(0)†	0.000	0.000	0.022	249.36	148.77	2.081	24.563	74
NA	0.879	(0)†	0.000	0.000	0.001	171.11	71.7	5.092	30.485	258

8.6.1 Auto Regressive

We apply the proposed approach to the AR detection model. In Table 8.2 we present the results of the WBC attack and compare them with the result from prior work black-box attacks [P3], while Figure 8.3 shows the impact of the WBC over the CUSUM statistics.

The AR detector precision and recall drop to 0 after the attack, this means that no more true positives are detected, and consequently, the F1 score becomes not defined as we have a division by zero. This result means that the detector is no longer capable of recognizing anomalies in the system. Looking at Figure 8.3 we can also observe the difference between the three attack approaches (baseline, NTP, NA). WBC baseline brings the CUSUM error to 0 when the ground truth label reports 'anomalous', this happens because the attacker iterates until the loss is minimized. This is in contrast to WBC NTP and WBC NA for which the attacker stops iterating as soon as the alarm threshold is not surpassed anymore. Finally, we can notice the difference between the WBC NTP and WBC NA, the WBC NTP (as the name suggests) brings the True Positives to zero, while the WBC NA hides all the positives (both True Positive and False Positive).

Regarding the computational time, we observe that the WBC concealment attacks required hundreds of milliseconds to compute (while SWaT sampling time is 1 second). WBC baseline is sensibly faster because code optimization was used. As in the WBC baseline we care of loss minimization, and we attack the AR model, we can achieve loss minimization in one step by selecting $\epsilon = ||\nabla_x Loss(x, o)||_2$ in Equation 8.2. For clarity, this is equivalent to changing Equation 8.2 to $\delta = -\nabla_x Loss(x, o)$.

We compare the white-box concealment technique w.r.t. the black-box attacks proposed by Erba et al. [P3] (See Table 8.2). As we can observe the white-box attacks outperform the black-box attacks in terms of concealment capability, as the black-box attacks never conceal all the True Positives (i.e. recall greater than 0). Finally, we can compare the Euclidean distances between the attacks. As we can observe in Table 8.2, the average perturbation is always lower for the WBC attacks w.r.t. prior work black-box

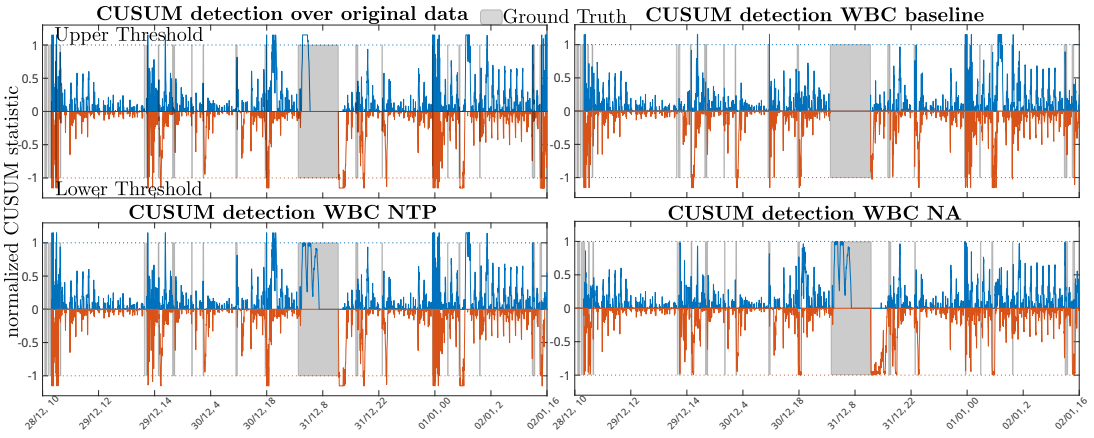


Figure 8.3: Comparison of AR detection before and after the WBC attack. The concealment attack hides the anomalies in the process data. In the bottom figure (WBC NA) WBC is applied to all the readings even if no physical attack is present, this removes not only the True Positives but also the False Positives.

Table 8.3: WBC Attack on the LTI model trained over SWaT.

						Elapsed(ms)		Euclidean D.			Ham. D.	
Data	Acc	F1	Prec	Rec	FPR	μ	σ	μ	σ	N	μ	σ
Original	0.962	0.815	0.987	0.694	0.001	-	-	-	-	-	-	-
Prior W. [P3]												
Replay	0.879	0.008	0.233	0.004	0.002	-	-	69.60	210.7	53863	21.4	1.8
Random R.	0.998	0.992	0.987	0.996	0.002	-	-	69.58	210.53	53863	21.4	1.8
Stale	0.887	0.126	0.845	0.068	0.002	-	-	66.05	211.53	53862	19.8	4.1
Our WBC												
baseline	0.884	0.081	0.785	0.043	0.002	121.0	326.1	67.25	218.92	53863	2.9	0.3
NTP	0.885	0.087	0.831	0.046	0.001	32.9	22.9	59.41	208.37	37385	2.9	0.5
NA	0.885	0.087	0.944	0.046	0.000	87.8	46.4	59.89	208.95	37881	2.9	0.5

attacks. This is because the white-box setting optimizes the samples to be optimal w.r.t. the past observed process data. This is instead impossible for black-box attacks. This can be observed by looking at the number of modified values (N) in Table 8.2, which is always in favor of the WBC NTP ad NA attacks. Since the AR model is univariate, we do not report the hamming distance (it would be 1 in any case).

8.6.2 Linear Time Invariant

We apply the WBC concealment attacks to the LTI model. Table 8.3 reports the results of our evaluation. The WBC concealment attacks evade the LTI detector, the detector recall drops from 0.69 to 0. We can observe the impact of the NA attack that reduces also the number of False Positive Rate.

The required computational time of the WBC attacks is at most 120 ms, which is lower than the sampling time of the SWaT system (1 second).

Also in this case the Euclidean Distance of the perturbed samples is lower than in

Table 8.4: WBC Attack on the SVM model trained over SWaT sensor LIT101, LIT301, LIT 401.

						Elapsed(ms)		Euclidean D.			Ham. D.	
Data	Acc	F1	Prec	Rec	FPR	μ	σ	μ	σ	N	μ	σ
Original	0.931	0.689	0.754	0.634	0.028	-	-	-	-	-	-	-
Prior W. [P3]												
Replay	0.855	0.0	0.0	0.0	0.028	-	-	87.13	266.40	53897	5.99	0.12
Random R.	0.855	0.0	0.0	0.0	0.028	-	-	87.45	266.03	53897	5.99	0.12
Stale	0.855	0.0	0.0	0.0	0.028	-	-	84.52	269.66	53896	5.84	0.55
Our WBC												
baseline	0.855	0.0	0.0	0.0	0.028	65.56	56.25	7.54	27.33	53934	5.99	0.07
NTP	0.855	0.0	0.0	0.0	0.028	103.84	33.13	7.54	27.33	34187	5.99	0.07
NA	0.880	0.0	0.0	0.0	0.000	107.22	61.49	10.71	36.74	45361	5.96	0.33

prior work attacks. Moreover, when looking at the Hamming Distance, we can observe how the number of features to be manipulated decreases (2.93 vs 28.4). This happens because our WBC is constrained to manipulate the output of the model x_k but cannot operate on the input q_k (see Equation 5.11). This number tells us that an attacker which controls 3 out of the 25 features used by the model, can significantly reduce the classifier recall by reducing it from 0.694 to 0.046. The number N is lower for WBC NTP and NA attacks when compared to prior work, i.e. 37881 vs 53863. When we compare the evasion performance of the attacks, we can observe that the WBC approach has comparable performance to the Replay and Stale attacks in terms of reduction of the model recall.

8.6.3 SVM

Table 8.4 reports the results of the evaluation of the proposed attacks on the SVM model. The proposed WBC concealment attack evades the SVM model and the recall drops from 0.63 to 0 in all the considered settings. We can observe how the NA approach differs by bringing the FPR to 0. The average computational time is at most 107ms, which is lower than the sampling rate of the SWaT testbed. Since the detector is using the water tank levels measured at d timesteps of distance (π,π'), we constraint the adversarial example to modify only π' (3 features), this is consistent with our challenge C8.1.

When comparing the WBC attacks to prior work generic concealment attacks, we can observe that the Euclidean distance required by the WBC attack is lower, as well as the number of perturbed samples by the NTP and NA approaches. The Hamming distance remains almost the same, after d timestep of continuous attack (in our case 1 step) all the features in (π,π') are under the control of the attacker (6 features).

8.6.4 PASAD

In this section, we attack PASAD with our WBC approach. The results of the attack are summarized in Table 8.5. Also in this case, the attacks are successful and the performance of the detector is compromised, as the recall drops close to 0 in all the

Table 8.5: WBC results on PASAD trained on SWaT Dataset sensor LIT301 (used in the paper (9)). The WBC attacks evade PASAD. The WBC requires less than 4ms to compute. The Euclidean distance is smaller when compared to prior work attacks. Threshold 3×10^6.

						Elapsed(ms)		Euclidean D.		
Data	Acc	F1	Prec	Rec	FPR	μ	σ	μ	σ	N
Original	0.878	0.557	0.492	0.641	0.090	-	-	-	-	-
Prior Work [P3]										
Replay	0.822	0.118	0.145	0.100	0.080	-	-	13.664	48.693	53859
Random R.	0.819	0.083	0.106	0.069	0.079	-	-	13.341	47.829	53852
Stale	0.899	0.617	0.563	0.681	0.072	-	-	17.356	56.065	52013
Our WBC										
baseline	0.825	0.039	0.057	0.03	0.067	2.31	1.84	12.92	48.716	40962
NTP	0.818	0.008	0.011	0.006	0.072	3.91	7.95	8.265	94.526	24842
NA	0.870	0.004	0.025	0.002	0.012	2.6	2.44	10.431	48.302	33369

three considered attacks. Differently from the previous case, the recall does not reach exactly 0, this is because there are a few instances in which the WBC is not reducing enough the distance from the PASAD cluster center. Similar to the previous experiment, we can see the difference between the baseline, NTP, and NA approaches. Again we can observe how the FPR rate reduces in the case of the NA setting. This time it reaches 0.012 meaning that there are few false positives.

Looking at the computational time required, the WBC algorithm finds the adversarial examples in 2.3 ms which is lower that the SWaT sampling time of 1 second. In this case, optimizations cannot be performed in the baseline setting, the method by applying the Singular Value Decomposition projects the univariate sensor readings time series into a subspace and tracks the distance of the projected time series form the centroid of normal operations cluster. As explained with the research challenge C8.1, we assume we cannot change the whole time series sliding window but we manipulate just the last observation from the coming from the physical process. For this reason the attack evades the detector by changing one sample at a time. Eventually, if the attack continues, all the samples in the sliding window are under control of the attacker.

Finally, if we compare the performance of the white-box attacks w.r.t. black-box attack from prior work [P3], we can observe that also in this case the WBC attacks are more effective than the black-box attacks in terms of concealment performance as the WBC recall score is always lower than in the case of the three attacks black box attacks from prior work. Looking at the Euclidean distance (Table 8.5), we can observe that the WBC attacks are on average less expensive than the black-box attack. Looking instead at the number of modified rows (N) we can observe that the WBC attacks are always less expensive than prior work.

8.6.5 SFIG

We then apply our attack method to the SFIG detector (see Table 8.6). In this setting, the WBC baseline and NTP coincide, because the invariant-based detector triggers only

Table 8.6: Attack against the SFIG detector on the SWaT dataset. The WBC baseline and NTP coincide because in invariant-based detectors alarms can be triggered only if rules are contradicted.

						Elapsed(ms)		Euclidean D.			Ham. D.	
Data	Acc	F1	Prec	Rec	FPR	μ	σ	μ	σ	N	μ	σ
Original	0.958	0.793	0.950	0.681	0.005	-	-	-	-	-	-	-
Prior W. [P3]												
Replay	0.876	0.000	0.004	0.000	0.005	-	-	69.60	210.70	53863	28.4	5.3
Random R.	0.893	0.240	0.797	0.141	0.005	-	-	69.58	210.53	53863	28.4	5.3
Stale	0.881	0.080	0.544	0.043	0.005	-	-	66.05	211.53	53862	27.4	8.4
Our WBC												
base./NTP	0.876	0.003	0.040	0.002	0.005	256.2	34.5	0.136	0.459	36704	2.8	0.5
NA	0.880	$(0)^{\dagger}$	0.0	0.0	0.0	354.3	264.6	0.141	0.465	38643	2.8	0.6

when rules are contradicted (i.e., there is no loss to minimize). In this experiment, we consider attacks that deal with the 51 features of the SWaT dataset, as the detector considers them all together.

Also in this setting, the detector was evaded by the attacks reducing the performance of the detector from 0.68 to 0 in both the cases. Also here we can appreciate the difference induced in the false positive rate in the two attack settings, the NA setting leaves no false positives.

In the WBC baseline/NTP, we notice that the recall is not 0.000, this is because we noticed that there is an artifact in the detection rules which causes a contradictory set of rules. This means that applying our attack to fix the data to turn off the alarms, triggers another rule in contradiction. This makes it impossible to turn off the alarm in a row of data.

Looking at the computational time required by the attacks in this case, we are in the order of 200/300ms which is lower than SWaT sampling time. Regarding the Euclidean distance, from Table 8.6 we can observe that the WBC attacks are less expensive than the attacks from the black box attacks from prior work, the proposed WBC attacks are always 2 orders of magnitude closer to the original values, meaning that features need to be slightly modified to achieve the goal. Also, the number of modified rows N (as in the previous experiments) is smaller. In this multivariate setting, we can also measure the number of features that were modified by the attack (i.e., the Hamming distance). As we can observe in Table 8.6, out of the 51 features in the SWaT dataset, WBC attacks modify on average 2.8 features (maximum 7 features out of 51), while prior work attacks modify on average $\sim$30 features (maximum 37 features out of 51).

Finally, if we compare the performance of the WBC w.r.t. attacks from prior work [P3], we can observe that on one hand the WBC NTP have a similar performance to Replay and Stale attacks from prior work, but on the other hand, as we pointed out before the WBC NTP is overall cheaper in terms of features that are modified by the attack.

Table 8.7: Summary of findings on our white-box concealment attacks. '# Manipulated' refers to the number of features that need to be manipulated by the attacker.

Method	Attack works	# Manipulated	Computational Cost $\leq$ 1sec
AR	✓	1/1	249ms
LTI	✓	3/25	120ms
SVM	✓	3/6	107ms
PASAD	✓	1/1	4ms
SFIG	✓	3/51	360ms

8.7 Discussion and Conclusion about White-box Concealment Attacks

In this section, we discuss the answers to our research questions. In Table 8.7 we summarize our findings. With respect to question **RQ8.1**, we tested three variations of the proposed WBC attacks, over five different anomaly detection systems. To do so the attacker has to deal with challenge **C8.1** (i.e., manipulate only the last sensor value) and with challenge **C8.2** (i.e., transform to differentiable detectors which not use a loss function). As a result, we found that the evaluated detectors are vulnerable to white-box concealment attacks, i.e., for all the tested detectors, the recall score drops to 0 or very close to it. This result demonstrates that the proposed attack methodology can affect a wide range of anomaly detectors for cyber-physical systems, affecting their detection performance with often little perturbation of the sensor data (in terms of Hamming and Euclidean distance). Our analysis reveals that only a low number of resources need to be under the control of an attacker to subvert the classification outcome of the target anomaly detector. For example, for the LTI and the SFIG our results show that is enough to control $\sim$3 features of the multivariate detector to conceal attacks.

With respect to research question **RQ8.2**, we measured the time to compute the adversarial examples (worst case $\sim$350ms), and we found that runtime manipulations are possible, as it is possible to compute manipulations faster then system's sampling rate of the SWaT system (1 sample per second). We note that temporal constraints for adversarial examples are not generally investigated by related adversarial machine learning literature, as in other domains adversarial examples can pre-computed (for example in the image classification domain) and do not need to be adapted based on the context.

With respect to research question **RQ8.3**, we compared the proposed attacks with black-box attacks from prior work [P3], in particular in terms of concealment performance and Euclidean distance. We found that our proposed WBC attacks are more effective (e.g., F1 score of the PASAD model is always lower in the WBC attack 0.039 vs 0.083 from prior work). Moreover, in general, our attacks require less manipulation than prior work attacks, (e.g., the Euclidean Distance in the SFIG case is 0.136 vs 66.05 from prior work, same the holds for the Hamming distance WBC 2.81 vs 27.41 from prior work).

Our results demonstrate that it is possible to evade a wide range of detectors while reducing the number of samples that need to be manipulated (compared to prior black-

box concealment attacks). Those findings highlight the need for further research and constructive discussion about guarantees for CPS anomaly detectors against adversarial manipulation. As such we see our contribution toward the robustness and reliability of CPS detectors against adversarial examples.

9
Conclusion

We started this thesis by investigating the security of novel CPS protocols and architectures. First, we evaluated secure-by-design industrial protocol for Cyber-Physical Systems. We systematically uncovered security issues with the protocol implementations, allowing an attacker to impersonate other devices in the network and perform middle-person, rogue client, and rogue server attacks. Second, we evaluated the security of robotic vehicle architectures. We proposed a novel attack vector which we call Sensor Deprivation Attacks. The proposed attacks exploit unauthenticated sensor communication to reconfigure a sensor in the shared bus. We discussed the theoretical effects of such attacks on the control loop and verified them in emulation environments and hardware flight controllers and quadcopters. Finally, we investigated the controllability of the Sensor Deprivation Attacks by proposing a Reinforcement-learning attack synthesis framework.

The results presented in the first part of the thesis show that CPS security remains an open challenge. Novel solutions are being proposed but there are major challenges in their configuration that will result in insecure deployments. In emerging architectures, security is also challenging as communication protocols between low-level components do not employ secure mechanisms. The identified issues require interdisciplinary discussion to be addressed, as the CPS community brings together diverse backgrounds and requirements (e.g., real-time operating systems, security, systems engineering, mechanical engineering).

In the second part of the thesis, we propose the first systematic evaluation of the security of Anomaly Detection systems, which were introduced to be seamlessly integrated into the CPS system as a method to uncover abnormal behaviors during the system operations. To perform such analysis, we propose a taxonomy of related process-based anomaly detectors from the literature and define the attacker model, constrained to operate within the real-time Cyber-Physical System. We characterize the attacker's goal (Concealment Attack), in which the attacker aims to hide an ongoing anomaly from the detection system. We refer to Adversarial Machine Learning literature, to describe the attacker's goal and highlight which novel challenges our attacker faces compared to prior proposed evasion attacks. Based on this proposed attacker, we propose three frameworks to assess the robustness of a variety of process-based anomaly detectors proposed in prior related work. We first evaluate the security of reconstruction-based anomaly detectors. We show that under our constrained attacker model, replay attacks (usually considered the strongest attacks to evade detection [134]), are easily detected because they break physical correlations among features. We then propose two attacks, with different assumptions on the attacker's knowledge of the target system, and demonstrate how a constrained attacker can evade such anomaly detection systems. We continue our evaluation by proposing a framework to test if the anomaly detectors correctly abstract the physical properties of the CPS. We introduce the Generic Concealment attacks to verify three consistency properties (spatial, temporal, and statistical). Our results show that all the detectors considered fail to abstract at least one of the properties of the CPS. Based on the consistency properties, and our framework, we demonstrated how to construct a process-based anomaly detector that is at the same time detecting process anomalies and concealment attacks. Finally, we were interested in understanding the minimal perturbation required for an attacker to evade process-based anomaly detectors. To do so we employed white-box concealment attacks and adapted them to work on

detectors that do not optimize a loss function. Our results demonstrated that a limited number of features under the control of the attacker is enough to completely evade the anomaly detection system.

The results presented in the second part of the thesis demonstrate that the countermeasures proposed to detect CPS attacks can be evaded to conceal anomalies. We believe that the proposed methodologies and techniques enable a critical evaluation and discussion around anomaly detection techniques for CPS, that go beyond the performance metrics of the anomaly detectors on the dataset. Our results, for the first time, demonstrate that the measured anomaly detection performance obtained over a dataset of anomalies does not suffice to draw conclusions about the effectiveness of a detection methodology. Instead, a critical analysis involving the physical properties of the modeled process is fundamental to provide guarantees about a detection technique.

9.1 Future Directions

The first Kranzberg law of technology [110] states "Technology is neither good nor bad; nor is it neutral". This thesis continues the line of research on CPS security. Our findings highlight the need for critical evaluation of solutions proposed to defend cyber-physical systems, especially taking into account whether newly introduced security solutions represent an advancement or bring the risk of a false sense of security.

Initial key distribution for industrial equipment remains an open issue for CPS. In air-gapped systems, we cannot rely on certificate authorities as in web applications. Solutions to this issue might involve human aspects, i.e., the usability of industrial equipment and their secure deployment.

Regarding Robotic Vehicle architectures, the risk of using unauthenticated serial protocols in mobile systems is well-known and understood [117]. Solutions to such problems have been proposed already for a decade. But in practice, until today, little has been done to mitigate such risks, as the state-of-the-art sensor continues communicating over insecure channels. Adoption of such solutions incurs extra production costs (like the cost of shielding the microcontroller). In practice, we believe that novel solutions, that account for the device's resource constraints, need to be researched to achieve an effective adoption.

Instead, the adversarial control of the vehicles demonstrated in Chapter 4 is an open research problem. This requires developing techniques to evaluate the robustness of control algorithms in performing their task in the presence of an attacker (following the line of robust control). A CPS must meet control deadlines to guarantee the stability of the system. An attacker by systematically inducing instability in the system will force an alternate objective to the system. This challenge can be approached at different levels, applying different techniques and assumptions on the target CPS.

Finally, regarding anomaly detection technology, a common argument against the deployment of such solutions in the field is the number of false alarms that are raised during execution. The results presented in this thesis highlight the novel risk of concealment attacks by deploying this technology in the field. Being aware of such prior limitations and issues is the first step to research novel defenses that are both reliable (i.e., with low false positives) and robust (i.e., robust against concealment attacks).

Bibliography

Author's Papers for this Thesis

[P1] Erba, A., Taormina, R., Galelli, S., Pogliani, M., Carminati, M., Zanero, S., and Tippenhauer, N. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Austin, USA, Dec. 2020.

[P2] Erba, A., Müller, A., and Tippenhauer, N. Security analysis of vendor implementations of the opc ua protocol for industrial control systems. In: *Proceedings of the 4th Workshop on CPS & IoT Security and Privacy (CPSIoTSec '22)*. ACM, Los Angeles, CA, USA, Nov. 2022. **Best Paper Award**.

[P3] Erba, A. and Tippenhauer, N. Assessing model-free anomaly detection in industrial control systems against generic concealment attacks. In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, Austin, USA, Dec. 2022. **Distinguished Paper with Artifact Award**.

[P4] Erba, A. and Tippenhauer, N. White-box concealment attacks against anomaly detectors for cyber-physical systems. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer Nature Switzerland, Cham, 2023, 111–131. Reproduced with the permission from Springer Nature. **Best Paper Award.**

[P5] Erba, A., Catellanos, J. H., Sihag, S., Zonouz, S., and Tippenhauer, N. Sensor Deprivation Attacks for Stealthy CPS Manipulation. Paper in preparation for conference submission. 2023.

Other Papers of the Author

[S1] Turrin, F., Erba, A., Tippenhauer, N., and Conti, M. A statistical analysis framework for ICS process datasets. In: *Proceedings of the Joint Workshop on CPS&IoT Security and Privacy (CPSIOTSEC'20)*. ACM, Nov. 2020.

[S2] Walita, T., Erba, A., Castellanos, J. H., and Tippenhauer, N. Blind concealment from reconstruction-based attack detectors for industrial control systems via backdoor attacks. In: *Proceedings of the Cyber-Physical System Security Workshop (CPSS)*. 2023.

Other references

[1] ABB. *AC500 user management with V3configuration and handling, version 3ADR010315*. Nov. 2020. URL: https://library.e.abb.com/public/2745203d43f64bf2af9add792d5b4e46/AC500%20-%20Application%20Note%203ADR010315.pdf.

[2] Abbasi, A. and Hashemi, M. Ghost in the plc designing an undetectable programmable logic controller rootkit via pin control attack. *Black Hat Europe* 2016 (2016). Publisher: Black Hat, 1–35.

[3] Abrams, M. Malicious control system cyber security attack case study–Maroochy Water Services, Australia (2008).

[4] Acar, Y., Backes, M., Fahl, S., Garfinkel, S., Kim, D., Mazurek, M. L., and Stransky, C. Comparing the usability of cryptographic APIs. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, 154–171.

[5] Adepu, S. and Mathur, A. Distributed detection of single-stage multipoint cyber attacks in a water treatment plant. In: *Proceedings of the ACM ASIA conference on computer and communications security (ASIACCS)*. 2016.

[6] Ahmed, C. M., Adepu, S., and Mathur, A. Limitations of state estimation based cyber attack detection schemes in industrial control systems. en. In: *2016 Smart City Security and Privacy Workshop (SCSP-W)*. IEEE, Vienna, Austria, Apr. 2016, 1–5.

[7] Ahmed, C. M., Ochoa, M., Zhou, J., Mathur, A. P., Qadeer, R., Murguia, C., and Ruths, J. NoisePrint: Attack detection using sensor and process noise fingerprint in cyber physical systems. In: *Proceedings of the 2018 on asia conference on computer and communications security*. ASIACCS '18. ACM, New York, NY, USA, 2018, 483–497.

[8] Ahmed, C. M., Palleti, V. R., and Mathur, A. WADI: A water distribution testbed for research in the design of secure cyber physical systems. In: *Proceedings of the workshop on cyber-physical systems for smart water networks)*. ACM, 2017, 25–28.

[9] Aoudi, W., Iturbe, M., and Almgren, M. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. CCS '18. ACM, 2018, 817–831.

[10] Ardupilot. *Ardupilot*. Accessed June 2023. URL: ardupilot.org/ardupilot/index.html.

[11] Ardupilot. *Ardupilot: external emulation*. Accessed June 2023. URL: ardupilot.org/dev/docs/sitl-with-JSON.html.

[12] Ardupilot. *EKF Failsafe*. Accessed June 2023. URL: https://ardupilot.org/copter/docs/ekf-inav-failsafe.html.

[13] Ardupilot. *Extended Kalman Filter Navigation Overview and Tuning*. Accessed June 2023. URL: https://ardupilot.org/dev/docs/extended-kalman-filter.html.

[14] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., and Rieck, K. Dos and don'ts of machine learning in computer security. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, 3971–3988.

[15] ASNeG. *OPC UA implementation, version 3.8.1*. Dec. 2020. URL: https://github.com/ASNeG/OpcUaStack.

[16] B&R Industrial Automation GmbH. *ADI OPC UA server user's manual, version 1.27*. Nov. 2020. URL: https://www.br-automation.com/es-mx/descargar/industrial-pcs-and-panels/automation-pc-3100/kabylake-system-unit/windows-10-iot-enterprise-2016-ltsb/adi-opc-ua-server-users-manual/.

[17] Bachmann. *OPC UA client and server, version 09/2020*. Jan. 2021. URL: https://www.bachmann.info/fileadmin/media/Produkte/Vernetzung/Produktblaetter/Software_OPC-UA-client_server_en.pdf.

[18] Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. Can machine learning be secure? In: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. 2006, 16–25.

[19] Beckhoff. *TC3 OPC UA, version 2.9*. Jan. 2021. URL: https://download.beckhoff.com/download/document/automation/twincat3/TF6100_TC3_OPC-UA_EN.pdf.

[20] Beijer Electronics Automation AB. *OPC UA server - iX developer 2.4 KI00312A*. Nov. 2020. URL: https://www.beijerelectronics.tw/en/Support/file-archive-tree-page?docId=58657.

[21] Bhupathiraju, S. H. V., Sheldon, J., Bauer, L. A., Bindschaedler, V., Sugawara, T., and Rampazzi, S. Emi-lidar: uncovering vulnerabilities of lidar sensors in autonomous driving setting using electromagnetic interference. In: *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2023, 329–340.

[22] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In: *Machine learning and knowledge discovery in databases*. Ed. by Blockeel, H., Kersting, K., Nijssen, S., and Železný, F. 2013, 387–402.

[23] Biggio, B. and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84 (2018). Publisher: Elsevier, 317–331.

[24] Bosch Rexroth AG. *OPC UA server application manual, R911403778 edition 01*. Nov. 2020. URL: https://www.boschrexroth.com/various/utilities/mediadirectory/download/index.jsp?object_nr=R911403778.

[25] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[26] Brockwell, P. J., Davis, R. A., and Fienberg, S. E. *Time series: theory and methods: theory and methods*. Springer Science & Business Media, 1991.

[27] BSI. *Open platform communications unified architecture security analysis*. Tech. rep. Bundesamt für Sicherheit in der Informationstechnik, 2017.

[28] Cao, Y., Bhupathiraju, S. H., Naghavi, P., Sugawara, T., Mao, Z. M., and Rampazzi, S. You can't see me: physical removal attacks on lidar-based autonomous vehicles driving frameworks. In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, 2993–3010.

[29] Cao, Y., Wang, N., Xiao, C., Yang, D., Fang, J., Yang, R., Chen, Q. A., Liu, M., and Li, B. Invisible for both camera and lidar: security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, 176–194.

[30] Cao, Y., Xiao, C., Cyr, B., Zhou, Y., Park, W., Rampazzi, S., Chen, Q. A., Fu, K., and Mao, Z. M. Adversarial sensor attack on LiDAR-Based perception in autonomous driving. In: *Proc. of the ACM conference on computer and communications security (CCS)*. ACM, New York, NY, USA, 2019, 2267–2281.

[31] Cárdenas, A., Amin, S., Sinopoli, B., Giani, A., Perrig, A., and Sastry, S. S. Challenges for securing cyber physical systems. In: *Workshop on future directions in cyber-physical systems security*. DHS, July 2009.

[32] Cárdenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S. Attacks against process control systems: risk assessment, detection, and response. In: *Proceedings of the 6th ACM symposium on information, computer and communications security*. 2011, 355–366.

[33] Cárdnas, A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S. Attacks against process control systems: Risk assessment, detection, and response. In: *ACM symp. Inf. Comput. Commun. Security*. 2011.

[34] Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In: *Proc. of the IEEE symposium on security and privacy*. ISSN: 2375-1207. May 2017, 39–57.

[35] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[36] Carlini, N. and Wagner, D. Audio adversarial examples: Targeted attacks on speech-to-text. In: *2018 IEEE security and privacy workshops (SPW)*. IEEE, 2018, 1–7.

[37] Cervini, J., Rubin, A., and Watkins, L. Don't drink the cyber: Extrapolating the possibilities of oldsmar's water treatment cyberattack. In: *International conference on cyber warfare and security*. Vol. 17. 2022, 19–25.

[38] Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., and Kohno, T. Comprehensive experimental analyses of automotive attack surfaces. In: *20th USENIX security symposium (USENIX Security 11)*. 2011.

[39] Chen, C.-T. *Linear system theory and design*. 3rd. Oxford University Press, Inc., USA, 1998.

[40] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: *Proceedings of ACM workshop on artificial intelligence and security*. ACM, 2017, 15–26.

[41] Chen, Y., Poskitt, C. M., and Sun, J. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In: *Proc. of the IEEE symposium on security and privacy*. IEEE, San Francisco, CA, 2018, 648–660.

[42] Cherubini, M., Meylan, A., Chapuis, B., Humbert, M., Bilogrevic, I., and Huguenin, K. Towards usable checksums: Automating the integrity verification of web downloads for the masses. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, 1256–1271.

[43] Choi, H., Kate, S., Aafer, Y., Zhang, X., and Xu, D. Software-based realtime recovery from sensor attacks on robotic vehicles. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association, San Sebastian, Oct. 2020, 349–364.

[44] Choi, H., Lee, W.-C., Aafer, Y., Fei, F., Tu, Z., Zhang, X., Xu, D., and Xinyan, X. Detecting attacks against robotic vehicles: A control invariant approach. In: *Proc. of the ACM conference on computer and communications security (CCS)*. ACM, 2018, 801–816.

[45] CODESYS. *CODESYS OPC UA PubSub SL*. Jan. 2021. URL: https://store.codesys.com/op-ua-pubsub-sl.html?___store=en&___from_store=default.

[46] Codesys. *OPC UA server, version 3.5.14.0*. Dec. 2020. URL: https://help.codesys.com/api-content/2/codesys/3.5.14.0/en/_cds_runtime_opc_ua_server/.

[47] Converter Systems LLC. *OPC UA implementaion*. Dec. 2020. URL: https://github.com/convertersystems/opc-ua-client.

[48] Cui, J., Liew, L. S., Sabaliauskaite, G., and Zhou, F. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Networks* 90 (2019), 101823.

[49] Dahlmanns, M., Lohmöller, J., Fink, I. B., Pennekamp, J., Wehrle, K., and Henze, M. Easing the conscience with OPC UA: An Internet-wide study on insecure deployments. In: *Proceedings of the ACM internet measurement conference*. 2020.

[50] Dang, H., Huang, Y., and Chang, E.-C. Evading classifiers by morphing in the dark. In: *Proc. of the ACM conference on computer and communications security (CCS)*. ACM, 2017, 119–133.

[51] Dayanıklı, G. Y., Mohammed, A. Z., Gerdes, R., and Mina, M. Wireless manipulation of serial communication. In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 2022, 222–236.

[52] Dayanıklı, G. Y., Sinha, S., Muniraj, D., Gerdes, R. M., Farhood, M., and Mina, M. Physical-Layer attacks against pulse width Modulation-Controlled actuators. In: *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, Aug. 2022, 953–970.

[53] Dechand, S., Schürmann, D., Busse, K., Acar, Y., Fahl, S., and Smith, M. An empirical study of textual key-fingerprint representations. In: *25th {USENIX} security symposium ({USENIX} security 16)*. 2016, 193–208.

[54] *EarlyStopping*. Keras EarlyStopping callback. URL: https://keras.io/api/callbacks/early_stopping.

[55] Eaton. *XC300 modular control, version MN050005EN*. Dec. 2020. URL: https://es-assets.eaton.com/DOCUMENTATION/AWB_MANUALS/MN050005_EN.pdf.

[56] Eclipse Milo. *OPC UA implementation, version 0.5.1*. Dec. 2020. URL: https://github.com/eclipse/milo.

[57] Electric, S. *M580 BMENUA0100 OPC UA embedded module installation and configuration guide, version 10/2019*. Nov. 2020. URL: https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=PHA83350.00.pdf&p_Doc_Ref=PHA83350.

[58] Erba, A. *Evading anomaly detectors through concealment attacks: a study on industrial control systems*. Master Thesis, Politecnico di Milano, Italy. 2019.

[59] Fahl, S., Acar, Y., Perl, H., and Smith, M. Why Eve and Mallory (also) love webmasters: A study on the root causes of SSL misconfigurations. In: *Proceedings of the 9th ACM symposium on Information, computer and communications security*. 2014, 507–512.

[60] Faveri Tron, A. de, Longari, S., Carminati, M., Polino, M., and Zanero, S. Canflict: exploiting peripheral conflicts for data-link layer attacks on automotive networks. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Association for Computing Machinery, Los Angeles, CA, USA, 2022, 711–723.

[61] Feng, C., Li, T., Zhu, Z., and Chana, D. A deep learning-based framework for conducting stealthy attacks in industrial control systems. *arXiv preprint arXiv:1709.06397* (2017).

[62] Feng, C., Palleti, V. R., Mathur, A., and Chana, D. A systematic framework to generate invariants for anomaly detection in industrial control systems. In: *Proc. Network and distributed system security symp. (NDSS)*. 2019.

[63] Forster, V. *Drone Flies Lungs Between Hospitals For Transplant Patient*. 2021. URL: https://www.forbes.com/sites/victoriaforster/2021/10/14/drone-flies-lungs-between-hospitals-for-transplant-patient/?sh=140f5a3d6a5b.

[64] Foundation, O. *OPC unified architecture specification v1.04*. 2018. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture.

[65] Fournier-Viger, P., Lin, J. C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., and Lam, H. T. The SPMF open-source data mining library version 2. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2016, 36–40.

[66] Frank, J. Artificial intelligence and intrusion detection: current and future directions. In: *Proceedings of the 17th national computer security conference*. Vol. 10. Baltimore, MD. 1994, 1–12.

[67] Free OPC UA. *OPC UA implementation, version 0.98.12*. Dec. 2020. URL: https://github.com/FreeOpcUa/python-opcua.

[68] FreeOpcUa. *OPC UA C++ implementation*. Dec. 2020. URL: http://freeopcua.github.io/.

[69] French Ministry of Economy and Finance, Alliance Industrie du Futur, German Federal Ministry for Economic Affairs and Energy (BMWi), Plattform Industrie 4.0, Italian Ministero dello Sviluppo Economico and Piano Nazionale Impresa 4.0. *The structure of the administration shell: TRILATERAL PERSPECTIVES from france, italy and germany*. July 2021. URL: https://www.de.digital/DIGITAL/Redaktion/EN/Publikation/the-structure-of-the-administration-shell.pdf?__blob=publicationFile&v=3.

[70] Galloway, B., Hancke, G. P., et al. Introduction to industrial control networks. *IEEE Communications Surveys and Tutorials* 15, 2 (2013), 860–880.

[71] Garcia, L., Brasser, F., Cintuglu, M. H., Sadeghi, A.-R., Mohammed, O., and Zonouz, S. A. Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit. In: *Proceedings of the annual network & distributed system security symposium (NDSS)*. Feb. 2017.

[72] General Electric. *Certificate management and the OPC UA server, version 6.1*. Nov. 2020. URL: https://www.ge.com/digital/documentation/ifix/version61/Subsystems/OPCUASVR/content/opcua_about_the_trust_list.htm.

[73] Github Issue. *Regression with username/password authentication #244*. Jan. 2021. URL: https://github.com/eclipse/milo/issues/244.

[74] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, 249–256.

[75] Go opcua. *OPC UA implementation, version 0.1.13*. Dec. 2020. URL: https://github.com/gopcua/opcua.

[76] Goh, J., Adepu, S., Junejo, K. N., and Mathur, A. A dataset to support research in the design of secure water treatment systems. In: *International conference on critical information infrastructures security (CRITIS)*. Springer, 2016, 88–99.

[77] Goh, J., Adepu, S., Tan, M., and Lee, Z. S. Anomaly detection in cyber physical systems using recurrent neural networks. In: *High assurance systems engineering (HASE), 2017 IEEE 18th international symposium on*. IEEE, 2017, 140–145.

[78] Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Mądry, A., Li, B., and Goldstein, T. Dataset security for machine learning: data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 2 (2022), 1563–1580.

[79] Gong, C., Yang, Z., Bai, Y., Shi, J., Sinha, A., Xu, B., Lo, D., Hou, X., and Fan, G. Curiosity-driven and victim-aware adversarial policies. In: *Proceedings of the 38th Annual Computer Security Applications Conference*. ACSAC '22. Association for Computing Machinery, Austin, TX, USA, 2022, 186–200.

[80] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572 (2014).

[81] Govil, N., Agrawal, A., and Tippenhauer, N. O. On ladder logic bombs in industrial control systems. In: *Computer Security: ESORICS 2017 International Workshops, CyberICPS 2017 and SECPRE 2017, Oslo, Norway, September 14-15, 2017, Revised Selected Papers 3*. Springer. 2018, 110–126.

[82] Green, B., Krotofil, M., and Abbasi, A. On the significance of process comprehension for conducting targeted ICS attacks. In: *Proceedings of the 2017 workshop on cyber-physical systems security and PrivaCy*. 2017, 57–67.

[83] Green, M. and Smith, M. Developers are not the enemy!: The need for usable security APIs. *IEEE Security & Privacy* 14, 5 (2016). Publisher: IEEE, 40–46.

[84] Grosse, K., Papernot, N., Manoharan, P., Backes, M., and McDaniel, P. Adversarial examples for malware detection. In: *Proc. of the european symposium on research in computer security*. Springer International Publishing, 2017, 62–79.

[85] Hadžiosmanović, D., Sommer, R., Zambon, E., and Hartel, P. H. Through the eye of the PLC: Semantic security monitoring for industrial processes. In: *Proceedings of the 30th annual computer security applications conference*. ACSAC '14. ACM, 2014, 126–135.

[86] Halder, S., Ghosal, A., and Conti, M. Secure over-the-air software updates in connected vehicles: a survey. *Computer Networks* 178 (2020), 107343.

[87] Hau, Z., Demetriou, S., and Lupu, E. C. Using 3d shadows to detect object hiding attacks on autonomous vehicle perception. In: *2022 IEEE Security and Privacy Workshops (SPW)*. IEEE. 2022, 229–235.

[88] Hau, Z., Demetriou, S., Muñoz-González, L., and Lupu, E. C. Shadow-catcher: looking into shadows to detect ghost objects in autonomous vehicle 3d sensing. In: *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer. 2021, 691–711.

[89] Hau, Z., Kenneth, T., Demetriou, S., and Lupu, E. C. Object removal attacks on lidar-based 3d object detectors. In: *Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*. Vol. 2021. 2021, 25.

[90] Hau, Z. and Lupu, E. C. Exploiting correlations to detect false data injections in low-density wireless sensor networks. In: *Proceedings of the 5th on Cyber-Physical System Security Workshop*. 2019, 1–12.

[91] Hayes, M. A. and Capretz, M. A. Contextual anomaly detection framework for big sensor data. *Journal of Big Data* 2, 1 (2015). Publisher: Springer, 2.

[92] Hayes, M. H. *Statistical digital signal processing and modeling*. John Wiley & Sons, 2009.

[93] Hitachi. *HX series application manual, verison NJI-638(X)*. Nov. 2020. URL: https://www.hitachi-da.com/files/pdfs/produkte/SPS/HX/HX-CPU_Software(Tentative).pdf.

[94] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation* 9, 8 (1997). Publisher: MIT Press, 1735–1780.

[95] Honeywell International. *Control edge builder protocol configuration reference guide, version RTDOC-X288-en-161A*. Nov. 2020.

[96] Hottinger Baldwin Messtechnik GmbH. *OPC UA implementation*. Dec. 2020. URL: https://github.com/HBM/opcua.

[97] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. Adversarial machine learning. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011, 43–58.

[98] Illiano, V. P. and Lupu, E. C. Detecting malicious data injections in wireless sensor networks: A survey. *ACM Computing Surveys (CSUR)* 48, 2 (2015). Publisher: ACM, 24.

[99] Issue. *Client certificates are not validated #392*. Jan. 2021. URL: https://github.com/FreeOpcUa/python-opcua/issues/392.

[100] Issue. *Client certificates are not validated #44*. Jan. 2021. URL: https://github.com/ASNeG/OpcUaStack/issues/44.

[101] iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. *WADI datatset*. 2017.

[102] Jang, J., Cho, M., Kim, J., Kim, D., and Kim, Y. Paralyzing drones via emi signal injection on sensory communication channels. In: *Network and Distributed System Security Symposium (NDSS)*. 2023.

[103] Jeong, J., Kim, D., Jang, J., Noh, J., Song, C., and Kim, Y. Un-rocking drones: foundations of acoustic injection attacks and recovery thereof. In: *Network and Distributed System Security Symposium (NDSS)*. 2023.

[104] Kalman, R. E. A new approach to linear filtering and prediction problems (1960).

[105] Karthikeyan, G. and Heiss, S. PKI and user access rights management for OPC UA based applications. In: *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*. Vol. 1. IEEE, 2018, 251–257.

[106] Keliris, A. and Maniatakos, M. ICSREF: A framework for automated reverse engineering of industrial control systems binaries. *arXiv preprint arXiv:1812.03478* (2018).

[107] Khan, A., Kim, H., Lee, B., Xu, D., Bianchi, A., and Tian, D. (M2mon: building an mmio-based security reference monitor for unmanned vehicles. In: *USENIX Security Symposium*. 2021, 285–302.

[108] Kiran, R. U. and Reddy, P. K. Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: *Proceedings of the 14th international conference on extending database technology*. ACM, 2011, 11–20.

[109] Koubâa, A., Allouch, A., Alajlan, M., Javed, Y., Belghith, A., and Khalgui, M. Micro air vehicle link (MAVlink) in a nutshell: A survey. *IEEE Access* 7 (2019).

[110] Kranzberg, M. Technology and history:" kranzberg's laws". *Technology and culture* 27, 3 (1986), 544–560.

[111] Kravchik, M., Biggio, B., and Shabtai, A. Poisoning attacks on cyber attack detectors for industrial control systems. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. 2021, 116–125.

[112] Kravchik, M. and Shabtai, A. Detecting cyber attacks in industrial control systems using convolutional neural networks. In: *Proceedings of the 2018 workshop on cyber-physical systems security and PrivaCy*. ACM, 2018, 72–83.

[113] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. 2012, 1097–1105.

[114] Krombholz, K., Busse, K., Pfeffer, K., Smith, M., and Zezschwitz, E. von. "If HTTPS were secure, I wouldn't need 2FA"-End user and administrator mental models of HTTPS. In: *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, 246–263.

[115] Krombholz, K., Mayer, W., Schmiedecker, M., and Weippl, E. "I have no idea what I'm Doing"–On the usability of deploying HTTPS. In: *26th USENIX security symposium (USENIX security 17)*. 2017, 1339–1356.

[116] Krotofil, M., Cárdenas, A. A., Manning, B., and Larsen, J. CPS: driving cyber-physical systems to unsafe operating conditions by timing DoS attacks on sensor signals. In: *Proceedings of the 30th annual computer security applications conference*. ACM, 2014, 146–155.

[117] Kune, D. F., Backes, J., Clark, S. S., Kramer, D., Reynolds, M., Fu, K., Kim, Y., and Xu, W. Ghost talk: Mitigating EMI signal injection attacks against analog sensors. In: *Security and privacy (SP), 2013 IEEE symposium on*. IEEE, 2013, 145–159.

[118] Landen, M., Chung, K., Ike, M., Mackay, S., Watson, J.-P., and Lee, W. Dragon: deep reinforcement learning for autonomous grid operation and attack detection. In: *Proceedings of the 38th Annual Computer Security Applications Conference*. ACSAC '22. Association for Computing Machinery, Austin, TX, USA, 2022, 13–27.

[119] Lee, E. A. *Cyber physical systems: Design challenges*. Tech. rep. Issue: UCB/EECS-2008-8. EECS Department, University of California, Berkeley, Jan. 2008.

[120] Lenze. *Lenze OPC UA communication, version v1.1*. Nov. 2020. URL: https://download.lenze.com/AKB/English/201400321/Commissioning_Lenze_OPC_UA_V1_1.pdf.

[121] Li, S., Neupane, A., Paul, S., Song, C., Krishnamurthy, S. V., Chowdhury, A. K. R., and Swami, A. Stealthy adversarial perturbations against real-time video classification systems. *Proceedings of the Annual Network & Distributed System Security Symposium (NDSS)* (2019).

[122] LibUA. *OPC UA implmentation*. Oct. 2020. URL: https://github.com/nauful/LibUA.

[123] Lovisotto, G., Turner, H., Sluganovic, I., Strohmeier, M., and Martinovic, I. SLAP: improving physical adversarial examples with Short-Lived adversarial perturbations. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, 1865–1882.

[124] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In: *6th international conference on learning representations, ICLR 2018, vancouver, BC, canada, april 30 - may 3, 2018, conference track proceedings*. 2018.

[125] Maggio, M., Hamann, A., Mayer-John, E., and Ziegenbein, D. Control-system stability under consecutive deadline misses constraints. In: *32nd euromicro conference on real-time systems (ECRTS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2020.

[126] Maiti, S., Balabhaskara, A., Adhikary, S., Koley, I., and Dey, S. Targeted attack synthesis for smart grid vulnerability analysis. In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2023, 2576–2590.

[127] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[128] Marktscheffel, T., Gottschlich, W., Popp, W., Werli, P., Fink, S. D., Bilzhause, A., and Meer, H. de. QR code based mutual authentication protocol for Internet of Things. In: *2016 IEEE 17th international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*. 2016, 1–6.

[129] Mathur, A. and Tippenhauer, N. O. SWaT: A water treatment testbed for research and training on ICS security. In: *Proceedings of workshop on cyber-physical systems for smart water networks (CySWater)*. Apr. 2016.

[130] Meier, D., Patzer, F., Drexler, M., and Beyerer, J. Portable trust anchor for OPC UA using auto-configuration. In: *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*. Vol. 1. IEEE, 2020, 270–277.

[131] Melis, M., Demontis, A., Pintor, M., Sotgiu, A., and Biggio, B. Secml: A python library for secure and explainable machine learning. *arXiv:1912.10013* (2019).

[132] Miller, C. and Valasek, C. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015, S 91 (2015), 1–91.

[133] Mitsubishi Electric. *MELSEC iQ-R OPC UA server ModuleUser's manual, version RD81OPC96-SW1DND-ROPCUA-E*. Nov. 2020. URL: https://dl.mitsubishielectric.com/dl/fa/document/manual/plc/sh081693eng/sh081693engf.pdf.

[134] Mo, Y. and Sinopoli, B. Secure control against replay attacks. In: *Communication, control, and computing, 2009. Allerton 2009. 47th annual allerton conference on*. IEEE, 2009, 911–918.

[135] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 1765–1773.

[136] Mühlbauer, N., Kirdan, E., Pahl, M.-O., and Carle, G. Open-source OPC UA security and scalability. In: *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*. Vol. 1. IEEE, 2020, 262–269.

[137] Nassi, B., Bitton, R., Masuoka, R., Shabtai, A., and Elovici, Y. Sok: security and privacy in the age of commercial drones. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, 1434–1451.

[138] National Instruments. *InsightCM 3.7 manual*. Nov. 2020. URL: https://www.ni.com/documentation/en/insightcm/latest/serverconf/secure-opc-server/.

[139] Neu, C. V., Schiering, I., and Zorzo, A. Simulating and detecting attacks of untrusted clients in OPC UA networks. In: *Proceedings of the third central european cybersecurity conference*. 2019, 1–6.

[140] Node-opcua. *OPC UA implementation, version 2.13.0*. Dec. 2020. URL: http://node-opcua.github.io/.

[141] Oliynyk, D., Mayer, R., and Rauber, A. I know what you trained last summer: a survey on stealing machine learning models and defences. *ACM Computing Surveys* (2023).

[142] Omron. *NJ-series CPU unit OPC UA user's manual (W588)*. Nov. 2020. URL: https://assets.omron.eu/downloads/manual/en/v2/w588_nj-series_cpu_unit_opc_ua_users_manual_en.pdf.

[143] OPC Foundation. *OPC unified architecture specification part 12/section 7.7.7: GetRejectedList v1.04*. 2017.

[144] OPC Foundation. *OPC unified architecture specification part 4: Services v1.04*. 2017.

[145] OPC Foundation. *OPC unified architecture specification part 6: Mappings v1.04*. 2017.

[146] OPC Foundation. *OPC unified architecture specification part 2: Security model v1.04*. 2018.

[147] OPC Foundation. *OPC UA implemntation, version 1.4.363.49*. Dec. 2020. URL: https://github.com/OPCFoundation/UA-.NETStandard.

[148] opcua4j. *OPC UA implementation*. Dec. 2020. URL: https://code.google.com/p/opcua4j/.

[149] Open62541. *OPC UA implementation, version 1.1.2*. Dec. 2020. URL: http://open62541.org/.

[150] OpenScada UA Interface. *OPC UA implementation, version 1.7*. Dec. 2020. URL: http://oscada.org/main/documentation/.

[151] Ornaghi, A. and Valleri, M. Man in the middle attacks. In: *Blackhat conference europe*. Vol. 1045. 2003.

[152] Panasonic Electric Works. *HMWIN user manual, verison v208*. Nov. 2020. URL: https://mediap.industry.panasonic.eu/assets/download-files/import/mn_hmwin_user_peweu_en.pdf.

[153] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The limitations of deep learning in adversarial settings. In: *2016 IEEE european symposium on security and privacy (EuroSP)*. Mar. 2016, 372–387.

[154] Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In: *Proc. of the IEEE symposium on security and privacy*. ISSN: 2375-1207. May 2016, 582–597.

[155] Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. Sok: security and privacy in machine learning. In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, 399–414.

[156] Pierazzi, F., Pendlebury, F., Cortellazzi, J., and Cavallaro, L. Intriguing properties of adversarial ml attacks in the problem space. In: *Proc. of the IEEE symposium on security and privacy*. IEEE, 2020, 1332–1349.

[157] Polge, J., Robert, J., and Le Traon, Y. Assessing the impact of attacks on opc-ua applications in the industry 4.0 era. In: *2019 16th IEEE annual consumer communications & networking conference (CCNC)*. IEEE, 2019, 1–6.

[158] Quinonez, R., Giraldo, J., Salazar, L., Bauman, E., Cardenas, A., and Lin, Z. SAVIOR: Securing autonomous vehicles with robust physical invariants. In: *Proc. of the USENIX security symposium*. Boston, MA, Aug. 2020.

[159] *Reduce Learning Rate On Plateau*. URL: https://keras.io/api/callbacks/reduce_lr_on_plateau/.

[160] Rockwell Automation. *FactoryTalk linx gateway getting results guide, version FTLG-GR001C-EN-E*. Nov. 2020. URL: https://literature.rockwellautomation.com/idc/groups/literature/documents/gr/ftlg-gr001_-en-e.pdf.

[161] Rrushi, J. L. SCADA protocol vulnerabilities. In: *Critical infrastructure protection*. Springer, 2012, 150–176.

[162] Rust opc UA. *OPC UA implementation, version 0.7*. Dec. 2020. URL: https://github.com/locka99/opcua.

[163] S2OPC. *OPC UA implementation, version 1.1.0*. Dec. 2020. URL: https://gitlab.com/systerel/S2OPC.

[164] Sadeghi, A.-R., Wachsmann, C., and Waidner, M. Security and privacy challenges in industrial Internet of Things. In: *2015 52nd ACM/EDAC/IEEE design automation conference (DAC)*. 2015, 1–6.

[165] Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. Ml-leaks: model and data independent membership inference attacks and defenses on machine learning models. In: *Network and Distributed Systems Security (NDSS) Symposium 2019*. 2019.

[166] Sarkar, E., Benkraouda, H., and Maniatakos, M. I came, I saw, I hacked: Automated generation of process-independent attacks for industrial control systems. In: *Proceedings of the 15th ACM asia conference on computer and communications security*. 2020, 744–758.

[167] Sathaye, H., Strohmeier, M., Lenders, V., and Ranganathan, A. An experimental study of GPS spoofing and takeover attacks on UAVs. In: *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, Aug. 2022, 3503–3520.

[168] Sato, T., Shen, J., Wang, N., Jia, Y., Lin, X., and Chen, Q. A. Dirty road can attack: security of deep learning based automated lane centering under physical-world attack. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, 3309–3326.

[169] Schenato, L. To zero or to hold control inputs with lossy links? *IEEE Transactions on Automatic Control* 54, 5 (2009), 1093–1099.

[170] Schenato, L., Sinopoli, B., Franceschetti, M., Poolla, K., and Sastry, S. S. Foundations of control and estimation over lossy networks. In: vol. 95. 1. IEEE, 2007, 163–187.

[171] Schiller, N., Chlosta, M., Schloegel, M., Bars, N., Eisenhofer, T., Scharnowski, T., Domke, F., Schönherr, L., and Holz, T. Drone security and the mysterious case of dji's droneid. In: *Network and Distributed System Security Symposium (NDSS)*. 2023.

[172] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In: *International conference on machine learning*. PMLR. 2015, 1889–1897.

[173] Selvaraj, J., Dayanıklı, G. Y., Gaunkar, N. P., Ware, D., Gerdes, R. M., and Mina, M. Electromagnetic induction attacks against embedded systems. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 2018, 499–510.

[174] Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. CCS '16. ACM, 2016, 1528–1540.

[175] Shen, J., Won, J. Y., Chen, Z., and Chen, Q. A. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing. In: *Proc. of the USENIX security symposium*. Aug. 2020, 931–948.

[176] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, 3–18.

[177] Siemens AG. *OPC UA .NET Client for the SIMATIC S7-1500 OPC UA Server, version 109737901 V1.3*. Dec. 2020. URL: https://cache.industry.siemens.com/dl/files/901/109737901/att_955026/v3/109737901_OPC_UA_Client_S7-1500_DOKU_V13_en.pdf.

[178] Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M. I., and Sastry, S. S. Kalman filtering with intermittent observations. *IEEE transactions on Automatic Control* 49, 9 (2004), 1453–1464.

[179] Snouffer, E. Six places where drones are delivering medicines. *Nat Med* 28, 5 (2022), 874–5.

[180] Son, Y., Shin, H., Kim, D., Park, Y., Noh, J., Choi, K., Choi, J., and Kim, Y. Rocking drones with intentional sound noise on gyroscopic sensors. In: *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., Aug. 2015, 881–896.

[181] Stouffer, K., Falco, J., Scarfone, K., et al. Guide to industrial control systems (ICS) security. *NIST special publication* 800, 82 (2011), 16–16.

[182] Suciu, O., Marginean, R., Kaya, Y., Daume III, H., and Dumitras, T. When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks. In: *27th USENIX security symposium (USENIX security 18)*. 2018, 1299–1316.

[183] Sugawara, T., Cyr, B., Rampazzi, S., Genkin, D., and Fu, K. Light commands: Laser-Based audio injection attacks on Voice-Controllable systems. In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, 2631–2648.

[184] Sun, J., Cao, Y., Chen, Q. A., and Mao, Z. M. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In: *29th USENIX security symposium (USENIX security 20)*. USENIX Association, Aug. 2020, 877–894.

[185] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013).

[186] Taormina, R., Galelli, S., Douglas, H., Tippenhauer, N. O., Salomons, E., and Ostfeld, A. A toolbox for assessing the impacts of cyber-physical attacks on water distribution systems. Environmental Modelling Software. *Environmental Modelling Software* 112 (Feb. 2019), 46–51.

[187] Taormina, R. *AutoEncoders for Event Detection (AEED): a Keras-based class for anomaly detection in water sensor networks.* 2018.

[188] Taormina, R. and Galelli, S. A Deep Learning approach for the detection and localization of cyber-physical attacks on water distribution systems. *Journal of Water Resources Planning Management* 144, 10 (2018). Publisher: ASCE, 04018065.

[189] Taormina, R., Galelli, S., Tippenhauer, N. O., Salomons, E., Ostfeld, A., Eliades, D. G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M. K., Brentan, B. M., Campbell, E., Lima, G., Manzi, D., Ayala-Cabrera, D., Herrera, M., Montalvo, I., Izquierdo, J., Luvizotto Jr, E., Chandy, S. E., Rasekh, A., Barker, Z. A., Campbell, B., Shafiee, M. E., Giacomoni, M., Gatsis, N., Taha, A., Abokifa, A. A., Haddad, K., Lo, C. S., Biswas, P., Pasha M. Fayzul K.and Kc, B., Somasundaram, S. L., Housh, M., and Ohar, Z. The battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. *Journal of Water Resources Planning and Management* 144, 8 (Aug. 2018). Publisher: ASCE.

[190] The Register. *Software developer cracks Hyundai car security with Google search.* Aug. 2022. URL: `https://www.theregister.com/2022/08/17/software_developer_cracks_hyundai_encryption`.

[191] *Throwing star LAN tap.* URL: `greatscottgadgets.com/throwingstar/`.

[192] Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction APIs. In: *USENIX security symposium.* 2016, 601–618.

[193] Tu, Y., Rampazzi, S., Hao, B., Rodriguez, A., Fu, K., and Hei, X. Trick or heat? Manipulating critical temperature-based control systems using rectification attacks. In: *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security.* CCS '19. Association for Computing Machinery, New York, NY, USA, 2019, 2301–2315.

[194] Unified Automation. *Step-by-step connect example, version 1.4.2.* Jan. 2021. URL: `http://documentation.unified-automation.com/uaexpert/1.4.2/html/first_steps.html#connect_step-by-step`.

[195] Urbina, D., Giraldo, J., Cardenas, A. A., Tippenhauer, N. O., Valente, J., Faisal, M., Ruths, J., Candell, R., and Sandberg, H. Limiting the impact of stealthy attacks on industrial control systems. In: *Proceedings of the ACM conference on computer and communications security (CCS).* Oct. 2016.

[196] Urbina, D., Giraldo, J., Tippenhauer, N. O., and Cárdenas, A. Attacking fieldbus communications in ICS: Applications to the SWaT testbed. In: *Proceedings of singapore cyber security conference (SG-CRC)*. IOS Press, Singapore, Jan. 2016.

[197] Van Overschee, P. and De Moor, B. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica* 30, 1 (1994). Publisher: Elsevier, 75–93.

[198] WAGO Kontakttechnik GmbH. *WAGO-I/O-SYSTEM 750, 750-8xxx OPC UA server, version 1.0.0*. Nov. 2020. URL: https://www.wago.com/global/d/3597419.

[199] Walmart. *Walmart Drone Delivery by the Numbers*. Accessed June 2023. URL: https://corporate.walmart.com/newsroom/2023/01/05/walmart-drone-delivery-by-the-numbers.

[200] Wan, Z., Shen, J., Chuang, J., Xia, X., Garcia, J., Ma, J., and Chen, Q. A. Too afraid to drive: systematic discovery of semantic dos vulnerability in autonomous driving planning under physical-world attacks. In: *ISOC Network and Distributed Systems Security (NDSS) Symposium*. 2022.

[201] Weidmüller Interface GmbH. *u-create studio System manual, version 2696790000/02/04.2020*. Nov. 2020. URL: https://mdcop.weidmueller.com/mediadelivery/asset/900_91150.

[202] Weinberger, S. Computer security: Is this the start of cyberwarfare? *Nature* 174 (June 2011), 142–145.

[203] Wikipedia. *Amazon Prime Air*. 2023. URL: https://en.wikipedia.org/wiki/Amazon_Prime_Air.

[204] Wikipedia. *Colonial pipeline ransomware attack*. Last accessed on: 2022-05-21. URL: https://en.wikipedia.org/wiki/Colonial%5C_Pipeline%5C_ransomware%5C_attack.

[205] Williams, T. J. The Purdue enterprise reference architecture. *Computers in industry* 24, 2-3 (1994). Publisher: Elsevier, 141–158.

[206] Wu, X., Guo, W., Wei, H., and Xing, X. Adversarial policy training against deep reinforcement learning. In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, 1883–1900.

[207] Xie, Z., Yan, C., Ji, X., and Xu, W. Bitdance: manipulating uart serial communication with iemi. In: *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 2023, 63–76.

[208] Xu, W., Qi, Y., and Evans, D. Automatically evading classifiers. In: *Proceedings of the network and distributed systems symposium*. 2016.

[209] Yan, C., Xu, W., and Liu, J. Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle. *DEF CON* 24 (2016).

[210] Yokogawa. *OPC-UA Server (/E3)User's Manual, version IM 04L51B01-20EN*. Nov. 2020. URL: http://web-material3.yokogawa.com/IM04L51B01-20EN_010.pdf.

[211] Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., and Xu, W. DolphinAttack: Inaudible voice commands. In: *Proceedings of the ACM SIGSAC conference on computer and communications security (CCS)*. ACM, 2017, 103–117.

[212] Zhang, Y. and Rasmussen, K. Electromagnetic signal injection attacks on differential signaling. In: *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. 2023, 314–325.

[213] Zhou, C., Yan, Q., Shi, Y., and Sun, L. DoubleStar: Long-Range attack towards depth estimation based obstacle avoidance in autonomous systems. In: *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, Aug. 2022, 1885–1902.

[214] Zizzo, G., Hankin, C., Maffeis, S., and Jones, K. Adversarial attacks on time-series intrusion detection for industrial control systems. In: *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*. IEEE, Guangzhou, China, 2020, 899–910.

[215] ZVEI – German Electrical and Electronic Manufacturers' Association Automation Division. *Status report reference architecture model industrie 4.0 (RAMI4.0)*. July 2021. URL: https://www.zvei.org/fileadmin/user_upload/Themen/Industrie_4.0/Das_Referenzarchitekturmodell_RAMI_4.0_und_die_Industrie_4.0-Komponente/pdf/5305_Publikation_GMA_Status_Report_ZVEI_Reference_Architecture_Model.pdf.