Formal Threat Analysis of Machine Learning-Based Control Systems: A Study on Smart Healthcare Systems

Nur Imtiazul Haque[a,*], Mohammad Ashiqur Rahman[a,*], Selcuk Uluagac[b]

[a] *Analytics for Cyber Defense (ACyD) Lab, Florida International University, USA*
[b] *Cyber-Physical Systems Security Lab, Florida International University, USA*

Abstract

Modern cyber-physical systems (CPSs) use the Internet of Things (IoT) to collect and exchange data efficiently, monitor device/sensor level interaction effectively, and adopt new standards effortlessly. Machine learning (ML) models are growingly used in the controllers of these IoT-enabled CPSs for pattern identification, state estimation, prediction, and anomaly detection. However, sophisticated adversaries can launch various attacks on the communication network and the hardware/firmware to introduce corrupted sensor measurements to manipulate the ML-based CPSs and create critical physical hazards. Hence, analyzing the threat space of a CPS is essential to understand the system's strength and identify the most vital resources to protect. However, existing studies have not proposed any verifiable solution for the threat analysis of ML-based CPSs. This paper presents a novel framework that uses efficient mechanisms to extract constraints from ML-based decision models and perform a formal threat analysis to identify potential false data injection (FDI) attack paths and corresponding effects on an IoT-enabled ML-based CPS. Our framework can provide us with all possible attack vectors, each representing a set of sensor measurements to be altered for a CPS given a specific set of attack attributes. The attack vectors enable us to assess the system's resiliency, thus providing insight to enhance the system's robustness. We consider an internet of medical things-enabled safety-critical CPS naming smart healthcare system (SHS) as the reference case. We validate our framework on a real SHS dataset, proving our framework's success in revealing feasible FDI attack paths. Our evaluation using synthetic and two real SHS datasets also affirms the tool's efficacy in the threat analysis of ML-based CPSs.

Keywords: Cyber-physical systems, internet of things, machine learning, cyberattacks, threat analysis, formal model.

1. Introduction

The rise of the Internet of Things (IoT) has made the world experience a tremendous revolution in computation and communication, enabling almost every device, irrespective of its size, to connect to the internet. The ubiquity of IoT networks in the industrial domain has improved cyber and physical integration and made the systems more automated. In a standard IoT-networked cyber-physical system (CPS), various interconnected components, including sensing and computing devices, identify physical conditions and transmit information to a control server [1]. The controller located within the control server generates control signals to actuate the system's physical components. A notable example of the modern IoT-enabled CPS is the smart healthcare system (SHS). Safety-critical systems such as SHSs pose significant implications and challenges for IoT-incorporated systems, leading to the development of a distinct IoT subset dedicated to interconnected healthcare systems known as the Internet of Medical Things (IoMT). The IoMT has the potential to revolutionize healthcare by lowering costs, enhancing service quality, and providing personalized medical care for a wide range of individuals, including those with limited financial resources and

*Corresponding author

Table 1: Summary of the Comparative Analysis of Formal Modeling, Adversarial ML, and RL Approaches

Criteria	**Formal Analysis**	**Adversarial ML**	**RL**
Verifiable threat detection	✓	X	X
Solution identification guarantee	✓	X	X
Convergence	✓	✓	X
Suboptimal solution from large time-series models	X	X	✓
Computational efficacy	X	✓	✓

those residing far from major medical facilities. [2]. For ensuring real-time medication and treatment, most IoMT-based SHSs continuously collect data from wireless body sensor devices (WBSDs) and process them to make required control decisions for triggering implantable medical devices (IMDs). SHSs minimize treatment imprecision, expenses, delays, and patient mortality by reducing hospital admission while increasing effectiveness and dependability in treatment [3]. The recent pandemic (i.e., COVID-19) has made the need for SHSs more evident. A report states that 48% of Americans have experienced either delayed or denied medical care during the pandemic period. The health condition of 11% of them deteriorated due to this treatment latency [4].

The insights and hidden patterns from the massive amount of data generated by the IoT sensors are largely demystified by leveraging state-of-the-art machine learning (ML) algorithms. Accordingly, ML models enable automated responses, enhanced decision-making, anomaly detection, and the projection of future trends [5]. For example, adopting ML-based control systems increases efficiency, accessibility, and personalization in SHSs [6]. The introduction of ML in IoMT-enabled SHSs has enhanced the reliability of remote patient monitoring, increased the efficiency of the medical sensors, and eliminated latency between disease detection and medication. Although ML models are advancing CPS controllers to capture critical relationships among sensor measurements, there still exist numerous threats of stealthy attacks on IoT-enabled CPSs [7]. Moreover, the attack surface is increasingly growing due to the continuous expansion of the network and sensing devices, demanding more endpoints to protect [8]. Furthermore, small IoT sensor devices' computation power cannot deploy strong security/cryptographic solutions. Recent literature has shown that SHSs or smart medical devices can be exploited with a broad group of attacks, including hardware Trojans, malware (e.g., Medjack), Sybil, denial of service (DoS), man-in-the-middle (MITM) attacks [9]. Among them, MITM attacks are the most threatening ones since they allow adversaries to launch false data injection (FDI) attacks to alter IoMT sensor measurements and thus can create life-threatening situations. We can also see instances of FDI attacks in real life. Hackers have attempted to compromise IoT thermostats to adversarily raise the smart home temperature by exploiting vulnerabilities in the Heatmiser thermostat system and Google Nest platform [10, 11]. Furthermore, real-life FDI attacks have been launched targeting Google Nest Cam baby monitor and Amazon Ring doorbell for delivering threats and racial slurs [12, 13]. The vulnerabilities associated with IoMT-enabled SHSs are also evident from a medical device report in 2019 by Sensato, stating that an average hospital room has around 15-20 connected/IoMT devices with 6.2 (out of 10) cybersecurity vulnerabilities [14]. A recent security study found a significant vulnerability in an SHS, where an attacker can exploit a negative pressure room to leak deadly pathogens, posing a life-threatening risk to the patients. They achieved this by injecting false audio signals into the songs played by the facility's CCTV speakers, bypassing heating, ventilation, air conditioning, and room pressure monitoring systems [15]. Hence, it is imperative to study and analyze the security and resiliency of an SHS.

Formal analysis and ML (i.e., adversarial ML and reinforcement learning) demonstrate promising performance in threat identification from ML-based CPSs. The formal analysis is an efficient and reliable way of synthesizing provable attack vectors [16, 17]. Some frameworks also formally modeled rule-based IoT-enabled CPSs to analyze threat space [18, 19, 20]. However, the frameworks cannot examine potential threats from ML-based IoT-enabled CPSs since the rules/constraint acquisition process from the ML models is not straightforward. Some works have used formal analysis for verifying ML models [21, 22, 23]. Nonetheless, they are limited to verifying diverse ML models rather than synthesizing attacks by analyzing those models. Several research studies used adversarial ML-based techniques to synthesize attack samples with variable

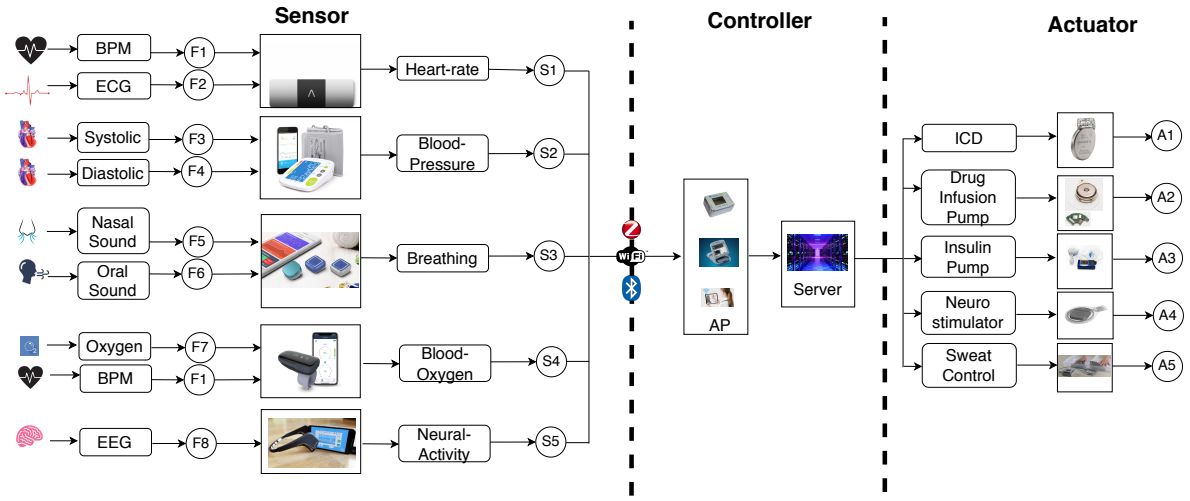


Figure 1: Sensor behavioral model in SHS.

attacker's capability. Adversarial ML compromises ML models by using methods that intentionally create unique input data called "adversarial examples." The purpose is to make the model give wrong outputs or behave incorrectly. These methods take advantage of how sensitive ML models are to small changes in input data. The input data is slightly altered so humans cannot perceive the exploitation, although the model produces erroneous output. This process involves making small changes in the input data to cause significant changes in what the model predicts. The major disadvantage of the adversarial ML-based approaches is that they intend to fool the human eyes. However, they cannot evade ML-based anomaly detection models that learn the benign data patterns. Although adversarial ML is substantially faster than formal techniques in identifying attack vectors, it cannot find verifiable attack vectors and does not guarantee identification even if one or more exist. The advantages of RL in threat analysis have been significantly highlighted in current research. In RL, agents are trained to detect attack vectors in ML models by engaging with a given environment. These agents monitor the system, act based on their observations, and receive feedback. Often, RL can achieve results faster and with better efficiency than traditional methods. Although not always guaranteed, RL may be linked with ML models to enhance attack vector identification. Nevertheless, RL has certain limitations as compared to formal approaches. Firstly, while RL-based approaches attempt to constrain state transitions by setting rewards or penalties, the learned model is not guaranteed to conform to these constraints strictly. In contrast, satisfiability modulo theories (SMT)-based solvers ensure compliance with set constraints and offer verifiable results. Moreover, while formal solvers can be time-consuming, they assure a solution if one exists. RL, on the other hand, doesn't offer such guarantees. Furthermore, RL often settles for suboptimal results, whereas formal methods aim for optimal outcomes. Lastly, determining the convergence of an RL-based model is still a challenging research problem. Where the complexity and magnitude of the problem surpass the capabilities of formal solvers, approaches rooted in RL are the primary alternative. Moreover, when there are stringent time restrictions, RL-based methodologies might generate solutions that could be more optimally efficient. In contrast, formal methods might be incapable of producing any solution. The summary of the comparative analysis is provided in 1.

This paper presents a security analysis framework that applies formal threat synthesis to identify potential attacks/threats on ML-based CPSs. In particular, the proposed formal framework performs threat analysis of SHSs that use ML-based disease classification models (DCMs) to deliver real-time treatment. It finds attack vectors in an SHS, each representing a set of sensor measurements within an adversary's access and resource capabilities that can be altered to change the disease classification/diagnosis. The rest of the paper will refer to this smart healthcare security analyzer as *SHChecker*. Furthermore, failures in safety-critical CPSs like SHSs can raise the possibility of life-threatening events. SHSs often leverage data validation using ML-based anomaly detection models (ADMs) [24] to minimize these incidents and increase dependability. These ADMs usually learn the pattern of inter-sensor measurement relationships by analyzing benign data. Our

threat analysis framework considers identifying stealthy attack paths that can bypass such data validation systems. SHChecker can assess potential attack vectors of an SHS that uses ML algorithms, e.g., decision tree (DT) [25], logistic regression (LR) [26], neural network (NN) [27] for classifying diseases and K-means clustering [28], density-based spatial clustering of applications with noise (DBSCAN) [29] for detecting anomalies. Our work mainly focuses on identifying critical FDI attack vectors and corresponding impacts that require minor alterations in fewer sensor measurements of an SHS. We verify our model's efficacy using the University of Queensland Vital Signs (UQVS) [30], PIMA Indians Diabetes datasets (we will refer to it as Diabetes dataset throughout the paper) [31], and a synthetic dataset named HealthGuard [32] through various performance metrics. In one of our recent works, we proposed a framework (i.e., SHATTER) for identifying attack vectors in occupants' activity-aware ML-based smart home systems [33]. However, the problem scope differs from the one considered in this work. SHATTER can identify attack vectors from time-series ADMs, unlike SHChecker. However, SHATTER is not designed to identify attack vectors from both DCM and ADM. Another significant advantage of SHChecker is that it can analyze ADM with more than two features, unlike SHATTER. Moreover, one of the features (i.e., only two features can be supported) of SHATTER-considered ADM needs to be a discrete type. However, SHChecker can work with all continuous features and any number of features if the number of constraints can be solved in a feasible time. In summary, our contributions are as follows:

- We formally model an SHS using first-order predicate logic by extracting constraints from the ML models to analyze the system critically.
- We develop a threat analysis framework (SHChecker) to identify potential attack vectors in ML-based CPSs (SHSs) by formally modeling FDI attacks with variable attack attributes. The source code can be found on GitHub [34].
- We conduct experiments with our formal threat analysis framework using real and synthetic SHS datasets to evaluate its performance in identifying critical sensor measurements and assessing the system's resiliency. We also evaluate the tool's scalability in analyzing the attack vectors.

The rest of the paper is organized as follows: we provide an overview of an SHS and its vulnerabilities, our proposed framework's attack model, and other necessary background information in Section 2. In Section 3, we present an overview of the proposed SHChecker framework and discuss its technical details, which include the constraint generation from an ML-based model and a formal model for threat analysis considering the adversary's knowledge, goal, capability, and accessibility. We also provide example case studies in section 5. The considered SHS ML models are presented in Section 4. We evaluate our proposed framework by running experiments using real and synthetic datasets and present the results in Section 6. A comprehensive literature review to highlight differences with the existing works is presented in Section 7. Section 8 summarizes the limitations and possible future extensions. Finally, we conclude the paper in Section 9.

2. System and Attack Model

This section provides an overview of IoMT-enabled SHSs and other related information to motivate the work and facilitate the readers' comprehension.

2.1. IoMT-based SHSs

IoT has been vastly adopted in most industrial CPSs. Therefore, several industries have been using different types of IoT networks to address specific domain challenges. For instance, SHS's connected smart medical device network is commonly known as the IoMT network. IoMT-enabled SHS is a game-changer for the medical field concerning consultation accuracy and cost reduction related to human labor. The enormous amount of medical data enables researchers to statistically analyze diseases and medication patterns. With the introduction of IoMT in the healthcare field, more attention has been paid to developing ubiquitous data accessing solutions to acquire and process data from decentralized data sources [35, 36, 37]. In the IoMT

Table 2: Example Devices and Parameters Considered for Monitoring Patients' Health Conditions

Vital Signs	Model	Feature Parameter Value	Database	Ref.
Heart Rate	KardiaMobile 6L	60-100 beats per minute	Fetal ECG Synthetic Database, Data.Gov	[40]
Blood Pressure	Greater Goods	Systolic (120 mm Hg) and Diastolic (80 mm Hg)	Fetal ECG Synthetic Database, Data.Gov	[40]
Blood Glucose	Dario	70-130 mg/dl	UCI ML database of diabetes	[41]
Blood Oxygen	iHealth Air Wireless Pulse Oximeter	Oxygen Saturation level $\geq$ 94%	Pattern Analysis of Oxygen Saturation Variability	[42]
Respiratory Rate	QuardioCore	12-20 Breaths per minute	BIDMC PPG and Respiration Dataset	[43]
Blood Alcohol	Scram Continuous Alcohol Monitoring	0.08 g/dl	StatCrunch dataset	[44]

network, data is often sent to a remote server to analyze and take control decisions due to the lack of processing limitations of medical sensors and IMDs. This paper considers an IoMT-enabled SHS incorporating WBSDs, ML model-assisted control systems, and IMD-based actuators. As demonstrated in Figure 1, an SHS can deliver medicine in real-time with a decision control system without requiring human involvement. In an SHS, patients are continuously monitored by the sensors attached to their bodies. These sensors deliver their observed measurement/s to the controller using various wireless communication protocols (e.g., WiFi, Bluetooth, Zigbee, and so on). The controller makes decisions based on the reported/received sensor measurements and sends control commands to the IMDs to deliver the necessary treatment to the patients. For example, Dario's blood glucose monitoring system seamlessly advertises blood glucose values to the controller (see Table 2). The controller checks whether the patient's vital signs are within normal ranges [38]. If the controller determines that the patient needs emergency insulin delivery, it notifies the responsible insulin pump implanted inside the patient's body to inject the proper amount of medication. A growing number of research attempts to develop closed-loop vital signs monitoring and drug delivery systems [39].

2.2. ML in SHS

As discussed, data-driven prediction accuracy and pattern/trend capturing capability have made ML algorithms pose significant implications in safety-critical CPSs, and healthcare is one of them [45]. In a growing industry of healthcare sensors that continuously gather a plethora of health data, the prevalence of using ML to analyze these data is gaining momentum. Our considered SHS uses a DCM and an ADM. The DCM uses a supervised ML model to label patient data accurately in real time. NN-based deep learning and rule-based ML models (e.g., DT) demonstrate significant performance in disease classification [46, 47]. The controller generates control signals based on the identified disease from the DCM. On the other hand, the ADM uses an unsupervised ML model to detect abnormal sensor measurements, learning the complex interrelation among them. The ADM verifies DCM-provided decisions.

The decision rules from the DCM are used to produce constraints associated with an SHS. However, to assess data validity using ADM-derived cluster constraints, we need to define the decision boundary formally. A concave hull algorithm has been used to formally model a tight bound for the clusters [48]. The concave hull algorithm often uses a k-nearest neighbor-based approach to fit the data points in a best-described polygon concave polygon that can be smoothed by a hyperparameter, k [49].

2.3. Cyberattacks in SHSs and our Attack Model

Malware and MITM attacks are predominant in SHSs. One of the most recent malware attacks, named "MEDJACK" (Medical Device Hijacking), exploits healthcare systems by placing malware within the IoMT networks [50]. MEDJACK is a stealthy cyberattack that utilizes the concept of polymorphic malware by constantly escalating its capability, making it very difficult to detect the attack. By creating a backdoor behind the firewall, MEDJACK gains access to the network without being detected [50]. MITM is a cyberattack where an adversary illegally gets into the communication between two authorized parties and eavesdrops on the transmitted data or corrupts it. Bluetooth-enabled medical devices exhibit potential vulnerabilities in sensor networks. Pournaghshband et al. [51] demonstrate the feasibility of launching an MITM attack in a Bluetooth-enabled pulse oximeter, which confirms that medical sensor measurements can be manipulated. They reverse-engineered a Nonin Onyx II 9550 fingertip pulse oximeter, which can measure

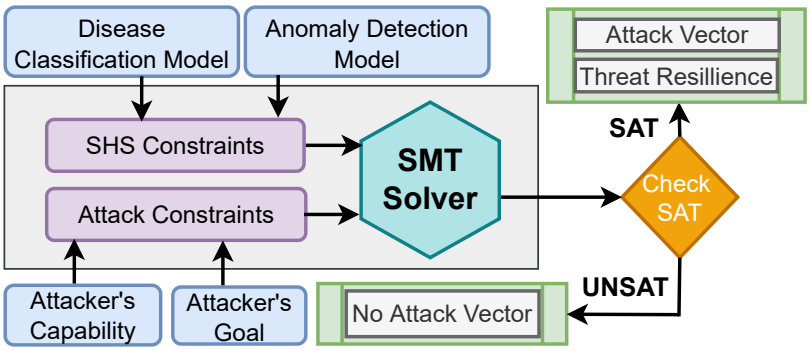


Figure 2: The workflow of SHChecker framework.

blood oxygen saturation and pulse rate. MITM attacks on wireless links can be performed in various ways, e.g., by jamming Bluetooth pairing with devices or access points (APs).

We are considering FDI or measurement manipulation attacks in our formal threat analysis framework. The sensors involved in an SHS IoMT network are typically energy-constrained, making them unable to implement strong encryption to ensure data integrity. As a result, attackers can find options to alter the measurement data. It is assumed that the attacker has access to one or several sensor measurements, and he/she can craft the measurements. The attacker can alter sensor measurements to lead the DCM to make an erroneous decision, thus making the controller send an improper control signal and delivering the wrong medication. The measurement alterations are considered to be performed intelligently to misclassify the patient status by the DCM to a different/targeted label without alarming the ADM. Our considered alteration scope is limited to the sensor level. The attack model assumes that the controllers and actuators in the SHS are secured/protected against being compromised directly. The attack model finds only those attack vectors that are attainable with the attacker's capability.

Various research approaches have validated the feasibility of FDI attacks in safety-critical CPSs. Two primary steps are mostly followed in launching stealthy FDI attacks- acquisition and alteration of sensor measurements. There are typically three types of nodes in an IoT network: nodes that send packets, forward packets, and nodes that gather packets and assess the consistency of each path's routing [52]. Through packet capture and analysis tools, the attacker with access to the router can play the role of an MITM and sniff the communication packets. An MITM attack also allows the attacker to alter/craft packet information by using ARP poisoning and IP/MAC addressing spoofing in addition to eavesdropping packets to alter the sensor measurement. It is difficult to avoid detection by the defense mechanism. Sensor measurement alteration can also be carried out by different means. A recent attack has shown the feasibility of sending inaudible voice commands from smartphones through malicious charging plugs and can send the wrong voice-controlled IoMT sensor measurements to the SHS controller [53]. A hazardous FDI attack impact is demonstrated in another research with real-world testbed implementation, where malicious music is used to create resonance in differential pressure sensors(DPSs), resulting in an overshooting of the DPS's normal pressure readings [15]. Wrong pressure readings can allow deadly pathogens to get leaked.

3. SHChecker Technical Details

The workflow of our proposed threat analysis framework, SHChecker, is presented in Figure 2. The framework first takes the DCM and the ADM model parameters. Then, the models get trained on an SHS dataset of various patient sensor measurements associated with corresponding patient status/labels. In this process, the DCM is trained to label the patient status, and the ADM is trained to verify the consistency of the sensor measurements for that specific label. Since the purpose of the proposed formal framework is to extract stealthy attack vectors from ML-based CPSs, it is necessary to formalize the attack vector identification as a constraint satisfaction problem (CSP), where the goal is to alter inputs (i.e., sensor measurements) to produce different label/output (i.e., targeted/untargeted) from the DCM. However, the

Table 3: Modeling Notations

Model Type	ML Model	Symbol	Description	Type
All	All	n_s	Number of sensors	Integer
		n_l	Number of labels	Integer
		a, b	Sensor measurement pair	Tuple
		j	Patient's status	Integer
		$\mathcal{S}$	Set of all sensors	Set
		$\mathcal{L}$	Set of all possible statuses for a patient	Set
Disease Classification Model (DCM)	Decision Tree	$\mathcal{N}^{DT}_{f,i}$	i-th node in path f	Integer
		$\mathcal{N}^{DT}$	Set of all nodes	Set
		$\mathcal{P}$	Set of all paths from root to leaf	Set
	Logistic Regression	$\theta_{g,i}$	Model parameter (weight) for i-th sensor and g-th patient status	Real
		θ	Set of model parameter (weight)	Set
		$\mathcal{B}^{LR}_g$	Model parameter (bias) for g-th patient status	Real
		$\mathcal{B}^{LR}$	Set of all model parameter (bias)	Set
	Neural Network	$\mathcal{N}^{NN}_{m,n}$	n-th node in m-th layer	Integer
		$\mathcal{N}^{NN}$	Sets of all nodes of all layers	Sets
		$\mathcal{W}_{m,o,n}$	Weight in the connecting link in between o-th node of (m - 1)th layer and n-th of m-th layer.	Real
		$\mathcal{W}$	Sets of all Weights.	Real
		$B^{NN}_{m,n}$	Bias in the node n from m-th layer.	Sets
		B	Set of all Biases.	Sets
Anomaly Detection Model (ADM)	DBSCAN, K-Means Clustering	$C_{a,b,j,k}$	k-th cluster for the sensor pair (a, b) for a patient with status, j	Cluster
		$\mathcal{C}_{a,b,j}$	Set of all clusters for the sensor pair (a, b) for a patient with status, j	Set
		$\mathcal{K}_{a,b,j,l}$	l-th line segment of the clusters for the sensor measurement pair (a, b) and the patient's status, j	Tuple
		$\mathcal{K}_{a,b,j}$	Set of all line segments of the clusters for the sensor pair (a, b) and the patient's status, j	Set
		$\mathcal{C}$	Sets of all clusters	Sets
		$\mathcal{K}$	Sets of all line segments	Sets

measurement alteration (attack vectors) should conform with the ADM to ensure stealthiness. Moreover, drastic alterations can produce irrational measurements (e.g., 150 mm Hg diastolic blood pressure) and/or create suspicion among the domain experts (e.g., healthcare professionals and security analysts) even though those evade the component ML models. Hence, the alteration scope to identify attack vectors is limited by both ML models and domain specifics. Furthermore, a set of measurements cannot be altered due to their high protection measures. Therefore, considering false measurement injections in the attack vector identification process is constrained/bounded. The constraints are abstracted or mathematically formulated by SHCHecker for attack vector extraction. The model parameters are leveraged to convert DCM and ADM into formal constraints. The constraints for the attack goal and the attacker's property/capability are formulated by analyzing the CPS domain. Finally, an SMT-based solver takes all the constraints associated with the SHS ML model and the attack constraints (attacker's capability, goal) as input. The SMT solver utilizes various background theories to solve the CSP and returns a satisfiable (SAT) or unsatisfiable (UNSAT) output. The SAT output from the solver implies that the given set of constraints is satisfied for the sensor measurements of the patient into consideration. At the time of SAT outcome, the framework reports an attack vector that includes a set of sensor measurements that needs to be attacked along with the minimum perturbation amount for misclassifying the considered patient's status based on the attacker's goal. An UNSAT result by the SMT solver signifies that the attack cannot attain the attacker's goal based on the given capabilities. The framework can identify the minimal attacker's capability requirement to produce a SAT outcome, which in turn provides insight into the system's resiliency. The framework increases the attacker's capability by providing them access to more resources, reexamining the attack feasibility, and repeating the process until it successfully finds an attack vector to identify the minimal capability requirement. The attack vector associated with minimal capability requirement implies that the attacker cannot succeed in launching any targeted/untargeted attack provided his accessibility to sensor measurements is less than the framework-reported minimal requirement. Hence, the system can be defined as *threat resilient* till that attacker's capability in terms of cybersecurity.

We have provided a generic overview of the framework. The detailed procedure of constraint derivation from the DCM (Section 3.1), ADM (Section 3.2) and attack model 3.3) are discussed as follows.

3.1. Derivation of DCM Constraints

The SHChecker framework consists of two major functional components: ML model constraints extraction mechanism and formal attack analysis model. Here, we discuss the formal constraints acquisition mechanism from different DCMs. As discussed, the DCM is a supervised ML model that labels a disease/health status for patients' vital sign/sensor measurements. Hence, the DCM can be considered a mathematical function that outputs patients' statuses based on input sensor measurements. Depending on the nature of the DCM, the mathematical function can be represented as a set of equalities and inequalities, which can be used as SHChecker's logical constraints. Such constraint formation is needed to specify the SHSs' behavior, as the DCM is a core part of it. During the attack space exploration by measurement alteration, the DCM constraints are required to guide and validate the satisfaction of the attack goal. Table 3 describes the notations we used in this paper for formal modeling.

DT Model Constraints: A DT algorithm returns an inference hierarchical rules-based model, from which formal model constraints acquisition to represent the model is quite straightforward. A boolean function $inference(\mathcal{S}, j)$ returns $True$ if sensor values are consistent with label j for DT inference rules.

DT contains several nodes starting from the root node. Each tree node consists of an attribute that denotes a sensor measurement using which the tree is split at that point with a threshold value. A patient's sensor measurement of that particular attribute having greater than the threshold follows the right path at that particular node. Otherwise, it follows the left path. This attribute and the threshold value generate a rule as shown in Equation 1.

$$rule(\mathcal{S}_a, d, e) = \begin{cases} \mathcal{S}_a \leq \mathbb{T}(d), & \text{if } e \text{ is a left node of } d \\ \mathcal{S}_a > \mathbb{T}(d), & \text{if } e \text{ is a right node of } d \end{cases} \tag{1}$$

Here, e is an immediate child of d and $a = \text{attr}(d)$. Figure 3 demonstrates a DT model for understanding the associated constraints. The tree is divided into multiple paths from the root to the leaf. Each path has a label and set of rules along its way. Equation 2 demonstrates the process of determining rules from a set of rules along a path.

$$rules(\mathcal{S}, f) = \bigwedge_{i=1}^{|\mathcal{N}_f^{DT}|-1} rule(\mathcal{S}_{attr(\mathcal{N}_{f,i}^{DT})}, \mathcal{N}_{f,i}^{DT}, \mathcal{N}_{f,i+1}^{DT}) \tag{2}$$

DT assigns a label j as patient status when sensor measurements associated with that patient satisfy all inference rules along a path having label j. Here, j is the label of the path's last node.

$$inference(\mathcal{S}, j) \rightarrow \exists_{f \in \mathcal{P}}(label(f) = j) \wedge rules(\mathcal{S}, f) \tag{3}$$

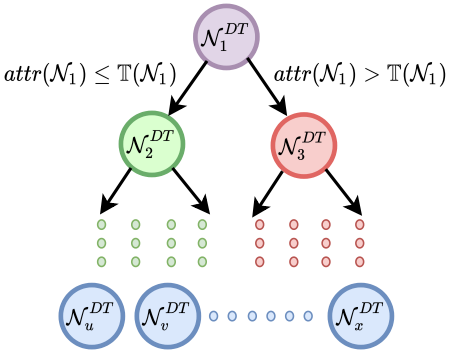


Figure 3: Flow diagram of DT model.

LR Model Constraints: An LR model assigns probabilities to patients' statuses based on the sensor measurements. After applying the softmax function, the label with the highest probability is selected as that patient status. In this case, the model parameters have been obtained by minimizing a cost function for optimal decision boundaries using maximum log-likelihood. Equation 4 shows the inference constraints for LR.

$$inference(\mathcal{S}, j) \rightarrow \arg\max_g \frac{exp((\sum_{i=1}^{n_s} \mathcal{S}_g \theta_{g,i}) + \mathcal{B}_g^{LR})}{\sum_{h=1}^{n_l} exp((\sum_{i=1}^{n_s} \mathcal{S}_h \theta_{hi}) + \mathcal{B}_g^{LR})} = j \tag{4}$$

NN Model Constraints: An NN comprises several layers, which can be categorized by an input layer, one or more hidden layers, and one output layer.

The input of each node at any layer except the input layer is calculated from the output, weights, and bias of the previous layer as demonstrated in Equation 5.

Suppose $N = |Nl|$, the number of layers in the model.

$$\forall_{m\in(1,N]} \quad input(\mathcal{N}_{m,n}) = \sum_{o=1}^{|\mathcal{N}_{m-1}|} (output(\mathcal{N}_{(m-1),o}) \times \mathcal{W}_{m,o,n}) + \mathcal{B}_m^{NN} \tag{5}$$

The input and output of layer 1 are the sensor measurement values as shown in Equation 6. Figure 4 demonstrates an NN model of N layers where the last hidden layer is denoted by ϵ (i.e., $\epsilon = N - 1$).

$$input(\mathcal{N}_1) = output(\mathcal{N}_1) = \mathcal{S} \tag{6}$$

For calculating the output of each node, input values of a particular node are passed through some complex activation function like relu, tanh, etc. It makes it simple for the model to generalize or adapt to a wide range of data and differentiate between outputs. Activation functions and their derivatives are primarily monotonic functions.

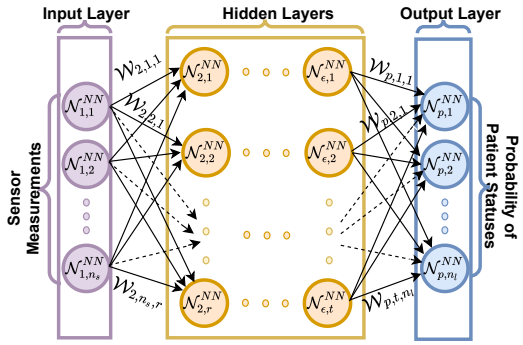


Figure 4: Flow diagram of NN model.

$$\forall_{m\in(1,N]} \quad output(\mathcal{N}_{m,n}) = activation(input(\mathcal{N}_{m,n})) \tag{7}$$

Label j is assigned to the patient in consideration if and only if the softmax function outcome of the jth output node gets a higher value than the other output nodes.

$$inference(\mathcal{S}, j) \rightarrow \arg\max_g \frac{exp(input(\mathcal{N}_{n,g})}{\sum_{q=1}^{n_l} exp(input(\mathcal{N}_{n,q})} = j \tag{8}$$

3.2. Derivation of ADM Constraints

The ADM is an unsupervised ML model that validates the consistency between sensor measurements. The ADM can be considered as a mathematical function that outputs a boolean result representing measurement consistency based on input sensor measurements. The ADM function can also be formulated as equalities and inequalities constraints. Constraint formation for the ADM is required to specify the SHSs' security properties. The ADM constraints are needed to validate the satisfaction of the attack's stealthiness. Constraints acquisition from ADMs is more challenging than DCMs due to their significantly higher nonlinear constraints. We develop an effective mechanism to devise constraints from DBSCAN and K-means clustering-based ADMs, as discussed below.

DBSCAN Model Constraints: To validate the consistency of a set of measurements, we use the DBSCAN model constraints. We consider consistency between the combination of all pairs of sensor measurements instead of sensor measurements to overcome the challenge of obtaining constraints in high dimensional space. This is because most clusters do not satisfy the requirement of constraint acquisition in high dimensional space due to the lack of sufficient cluster data points. We find these constraints according to the logical functions of checking if the measurements are within the clusters of that specific label. We explain this concept below with a simple example to facilitate the reader.

Figure 5 shows two clusters ($\mathcal{C}_{a,b,j,1}$ and ($\mathcal{C}_{a,b,j,2}$) in a 2D data plane where $\mathcal{C}_{a,b,j,1}$ consists of seven line segments ($\mathcal{K}_{a,b,j,1}$, $\mathcal{K}_{a,b,j,2}$, ..., $\mathcal{K}_{a,b,j,7}$) and $\mathcal{C}_{a,b,j,2}$ consists of three line segments ($\mathcal{K}_{a,b,j,8}$, $\mathcal{K}_{a,b,j,9}$, and $\mathcal{K}_{a,b,j,10}$). We denote the end points of any line segment ($\mathcal{K}_{a,b,j,i}$) are ($\mathcal{X}_{a,j,i}, \mathcal{Y}_{a,j,i}$) and ($\mathcal{X}_{b,j,i}, \mathcal{Y}_{b,j,i}$), where $\mathcal{Y}_{b,j,i} \geq \mathcal{Y}_{a,j,i}$. To validate the consistency of a data point (x, y), we use the following logical functions:

- *inRangeOfLineSegment* $(x, y, \mathcal{K}_{a,b,j,i})$: This function checks whether the point is within the vertical range of the line segment, $\mathcal{K}_{a,b,j,i}$.

$$inRangeOfLineSegment(x, y, \mathcal{K}_{a,b,j,i}) \rightarrow \mathcal{Y}_{a,j,i} < y \leq \mathcal{Y}_{b,j,i} \tag{9}$$

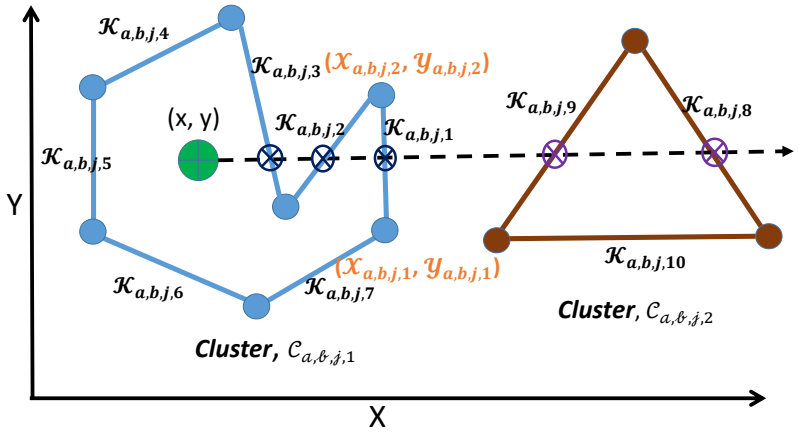


Figure 5: Logic behind checking if a point is inside a polygon cluster in DBSCAN algorithm.

Thus, according to Figure 5, we can say that $inRangeOfLineSegment(x, y, \mathcal{K}_{a,b,j,1})$ returns $True$ for a point (x, y) as y is within the range of $y_{a,b,j,1}$ and $y_{a,b,j,2}$.

However, for$\mathcal{K}_{a,b,j,4}$, $inRangeOfLineSegment(x, y, \mathcal{K}_{a,b,j,4})$ return $False$ as y is not in the vertical range.

- $leftOfLineSegment(x, y, \mathcal{K}_{a,b,j,i}$: This function checks if the point (x, y) is on the left side of the line segment, $\mathcal{K}_{a,b,j,i}$.

$$leftOfLineSegment(x, y, \mathcal{K}_{a,b,j,i}) \rightarrow (x(\mathcal{Y}_{a,j,i} - \mathcal{Y}_{b,j,i}) - y(\mathcal{X}_{a,j,i} - \mathcal{X}_{b,j,i}) - (\mathcal{X}_{a,j,i}\mathcal{Y}_{b,j,i} - \mathcal{X}_{b,j,i}\mathcal{Y}_{a,j,i})) < 0 \quad (10)$$

In the case of cluster $\mathcal{C}_{a,b,j,1}$ in Figure 5, $leftOfLineSegment(x, y, \mathcal{K}_{a,b,j,1})$ returns $True$ while $leftOfLineSegment(x, y, \mathcal{K}_{a,b,j,5})$ returns $False$.

- $intersect(x, y, \mathcal{K}_{a,b,j,i})$: An imaginary line is drawn from a point (x, y) to the right side, which is also parallel to the x axis. The function $intersect$ determines if the imaginary line intersects the line segment, $\mathcal{K}_{a,b,j,i}$. This intersection happens only when the point (x, y) is within the line segment's range and located on its left side. The function is formalized as follows:

$$intersect(x, y, \mathcal{K}_{a,b,j,i}) \rightarrow inRangeOfLineSegment(x, y, \mathcal{K}_{a,b,j,i}) \wedge leftOfLineSegment(x, y, \mathcal{K}_{a,b,j,i}) \quad (11)$$

From Figure 5, it is obvious that $intersect(x, y, \mathcal{K}_{a,b,j,1})$ returns $True$ only for line segments $1, 2, 3, 8$, and 9.

- $withinCluster(x, y, \mathcal{C}_{a,b,j,k})$: This function returns $True$ if the data point (x, y) is within the cluster, $\mathcal{C}_{a,b,j,k}$. For all the boundary line segments $\mathcal{K}_{a,b,j,i}$ of $\mathcal{C}_{a,b,j,k}$, the function calculates $intersect(x, y, \mathcal{C}_{a,b,j,i})$ and performs the XOR operation on them. If there are an odd number of intersections with the line segments, $withinCluster$ returns $True$ as the XOR operation on one or more $False$ values, and an odd number of $True$ values result in $True$. Thus, we define the function as:

$$withinCluster(x, y, \mathcal{C}_{a,b,j,k}) \rightarrow \bigoplus_{1 \leq i \leq |\mathcal{C}_{a,b,j,k}|} (intersect(x, y, \mathcal{K}_{a,b,j,i}) \wedge In(\mathcal{C}_{a,b,j,k}, \mathcal{K}_{a,b,j,i})) \quad (12)$$

Here, $In(\mathcal{C}_{a,b,j,k}, \mathcal{K}_{a,b,j,i})$ checks whether the line segment, $\mathcal{K}_{a,b,j,i}$ is from the cluster $\mathcal{C}_{a,b,j,k}$ or not. In Figure 5, (x, y) is inside cluster $\mathcal{C}_{a,b,j,1}$ as the imaginary line parallel to axis x from it intersects an odd number (three) of line segments of the cluster. On the other hand, the imaginary line intersects two line segments of cluster $\mathcal{C}_{a,b,j,2}$, and thus, we can conclude that the data point is outside of the cluster $\mathcal{C}_{a,b,j,2}$.

For formal modeling, let us assume that a total of n_s different sensor measurements are captured from a person's body, and they possess a health state from the n_l labels. Without the loss of generality, we assume that each measurement is recorded/reported by one sensor.

Besides, let the current label of the person be j, where $j \in \mathcal{L}$. To reduce the complexity of the clustering constraints, we consider the relationship of two sensors at a time. Thus, for the label $j \in \mathcal{L}$ and sensor pair $(a,\ b)$ where $(a,\ b) \in \mathcal{S}$ we get one or more clusters, $\mathcal{C}_k^{a,b,j}$, representing the relationship between the two measurements for that specific label. These clusters consist of a few line segments, which are represented as $\mathcal{K}_{a,b,j,l}$. To check the consistency of the data measurements with the constraints from DBSCAN, let us take a pair of two sensor measurements $\mathcal{S}_{a,b} = (\mathcal{S}_a, \mathcal{S}_b) \in \mathcal{S}$. We verify the consistency of measurement set $\mathcal{S}$ by checking if each pair of measurements is within any of the corresponding clusters, $\mathcal{C}_{a,b,j,k}$. The measurement set is consistent if the following conditions hold:

$$consistent(\mathcal{S}, j) \rightarrow \forall_{(a,b)\in\mathcal{S}\wedge(a!=b)}\exists_{k\in\mathcal{C}}\, withinCluster(\mathcal{S}_{a,b}, \mathcal{C}_{a,b,j,k}) \tag{13}$$

where,

$$withinCluster(\mathcal{S}_{a,b}, \mathcal{C}_{a,b,j,k}) \rightarrow \bigoplus_{l}(intersect(\mathcal{S}_a, \mathcal{S}_b, \mathcal{K}_{a,b,j,l}) \wedge In(\mathcal{C}_{a,b,j,k}, \mathcal{K}_{a,b,j,l})) \tag{14}$$

$$intersect(\mathcal{S}_a, \mathcal{S}_b, \mathcal{K}_{a,b,j,l}) \rightarrow inRangeOfLineSegment(\mathcal{S}_a, \mathcal{S}_b, \mathcal{K}_{a,b,j,l}) \wedge leftOfLineSegment(\mathcal{S}_a, \mathcal{S}_b), \mathcal{K}_{a,b,j,l}) \tag{15}$$

K-Means Clustering Model Constraints: Constraint acquisition from the K-means clustering model requires a similar approach as the constraint acquisition from the DBSCAN model. However, due to the algorithmic variation, the number of clusters and the number of noise points are different.

3.3. Formal Modeling of Attacks

The formal analysis lets us explore the search space of all possible system behaviors and figure out potential vulnerabilities. The formal attack constraints are devised from abstract modeling of FDI attack and attacker's property. The FDI attack constraint represents FDI attacks through mathematical expression, equalities, and logical functions. We consider the attacker's accessibility and capability as the attacker's property. The prior section has defined the ADM model formulation, which restricts the attacker's capability of arbitrary alteration. Here, we impose additional capability constraints to ensure stealthiness. The capability constraints are domain specifications that prevent generating irrational measurements and reduce the suspicion of domain experts. Depending on the security measures, some measurements can be considered unexploitable, formulated as attacker's accessibility constraints in our formal analysis. The proposed framework inputs the attack model (attacker's goal, attacker's accessibility, and capability) and the underlying system's model (ML model constraints), formally analyzes them, and finds out possible threats using an SMT-solver. The attacker's capability depends on the range of values that can be changed without alarming the system.

3.3.1. Attack Technique:

Our framework considers an adversary attempting to inject false sensor measurements into the original patient sensor measurements. The injected measurements are considered an attack vector if the attack goal is attained after injection.

$$\forall_{s\in\mathcal{S}}(\bar{\mathcal{S}}_s \rightarrow \mathcal{S}_s + \Delta\mathcal{S}_s) \tag{16}$$

Here, $\Delta\mathcal{S}$ denotes the vector containing injected sensor measurements, while $\bar{\mathcal{S}}$ indicates attacked sensor measurements.

3.3.2. Attacker's Accessibility:

An attacker may change a sensor measurement if they can access that particular measurement. The attacker cannot inject false measurements into the inaccessible sensor measurements.

$$\forall_{s \in \mathcal{S}}\ a_s \rightarrow (\Delta \mathcal{S}_s ! = 0) \tag{17}$$

Here a_s denotes the accessibility to sensor measurement, s.

3.3.3. Attacker's Capability:

The FDI attack success depends on the number of resources that can be modified and the alteration scope. Our formal framework specifies how many resources the attacker can access and alter.

$$\sum_{s \in \mathcal{S}} a_s \leq \mathbb{M} \tag{18}$$

$$\forall_{s \in \mathcal{S}}\ abs(\frac{\Delta \mathcal{S}_s}{\mathcal{S}_s}) < \mathbb{A} \tag{19}$$

Here, $\mathbb{M}$ limits the maximum number of sensors accessible by the attacker, and $\mathbb{A}$ denotes the allowed range of measurement alteration for achieving the attack goal without getting revealed. Even if compromised sensor measurements in a successful attack come from an existing cluster following all ADM constraints, drastic alteration (compared to recent values), we consider that the controller can identify drastic alteration of measurements. Hence, limiting the adversarial capability allows us to find stealthy attack vectors.

3.3.4. Attacker's Goal

In our work, we consider the attacker's goal to misclassify the patient's status and thus deliver the wrong medication. If the following constraint is satisfied, an attacker can change a patient's label from j to $\bar{j}$.

$$alter(j, \bar{j}) \rightarrow inference(\mathcal{S}, j) \wedge consistent(\mathcal{S}, j) \quad \wedge inference(\bar{\mathcal{S}}, \bar{j}) \wedge consistent(\bar{\mathcal{S}}, \bar{j}) \tag{20}$$

Equation 20 requires both the current and altered labels to be satisfied by the classification model and ADM constraints.

The flow of the overall framework functioning can be understood from the Algorithm 1. In the algorithm, we provided an intuitive overview of the complete process. The function $attackVectorExtraction$ takes the sensor measurements, sensor accessibility vector, thresholds for maximum allowable attacked measurement, exploitation range, and expected attack label (i.e., attack goal). First, we define several variables, including the attack vector (Lines 1 - 6). Then, we define a solver and add DCM and ADM constraints (Lines 7-9) as defined in the above equations. After that, we feed an input vector with the given sensor measurements (Line 11). Later, we restrain the solver from changing the measurements or adding false measurements only to the inputs that are accessible (Lines 14-20). We also constrain the measurement alteration range and the number of measurements that can be altered (Lines 28-30). Finally, we solve the constraints of the solver and get an attacked label. If the attacked label is consistent with the ADM and attack goal, the solver returns the attack vector; otherwise, it returns $UNSAT$ (i.e., constraints are not satisfiable).

4. SHS Modeling

Due to the sensitivity of data and system models, underlying ML models of SHS are hardly exposed to public research. We consider an IoMT-enabled SHS with two different types of ML models for threat analysis. Our proposed framework is designed to analyze threats from any ML-based systems' constraints, and in this section, we discuss the ML model constraint formulation processes. However, we are considering the best-performing ML model-based SHS among the considered models for evaluation purposes to identify the most critical threats. This section discusses choosing the best ML model (Section 4.1) and corresponding features for our considered SHS to be analyzed by SHChecker. Moreover, we have rationalized pair of feature selection for ADM (Section 4.2) and adoption of combined models (Section 4.3). We also discuss the pros and cons of different ML models in Section 4.1.

Algorithm 1: Attack Vector Extraction from SHCHecker Framework

```
Function attackVectorExtraction(𝒮, a, 𝕄, 𝔸, j̄):
    input ← []× length((S));
    attackedInput ← []× length((S));
    attackedLabel ← −1;
    attackVector ← []× length(a));
    measurementsAttacked ← []× length(a));
    Initialize a solver, s;
    s.addConstraints(dcmConstraints());
    s.addConstraints(admConstraints());
    for s in Range(𝒮) do
        s.addConstraints(input[s] ← 𝒮_s);
    end
    for i in Range(a) do
        if a[i] == 0 then
            s.addConstraints(attackVector[i] ← null);
            s.addConstraints(attackedInput[i] ← input[i]);
        end
        else
            s.addConstraints(attackedInput[i] ← attackVector[i]);
        end
        if attackVector[i] == null then
            s.addConstraints(measurementsAttacked[i] ← −1);
        end
        else
            s.addConstraints([i] ← 1;
        end
    end
    s.addConstraints(sum(measurementsAttacked) ≤ 𝕄;
    for s in Range(𝒮) do
        s.addConstraints(abs(attackVector[s] / input[s]) ≤ 𝔸);
    end
    attackedLabel ← f(s);
    if attackedLabel == j̄ and consistent(attackedLabel) then
        return attackVector;
    end
    else
        return UNSAT;
    end
return
```

4.1. ML Model for DCM and ADM

We train our DCMs and ADMs with optimal hyper-parameters to produce accurate and precise results. We use four different performance metrics to evaluate the performance of various ML models for SHS DCM: accuracy, precision, recall, and F1-score[54]. Accuracy calculates the number of correctly identified samples of overall data samples. Precision measures the false-positive rate, whereas recall quantifies the false-negative rate. F1-score takes precision and recall into account by performing harmonic mean of them. Table 4 shows that for both synthetic and generated datasets, DT works better than LR and NN based on accuracy, precision, recall, and f1-score.

For assessing the performance of considered ADMs, three different performance metrics have been considered. Table 5 presents a comparative analysis of DBSCAN and K-means clustering based on some internal cluster validation metrics naming average Silhouette Coefficient Score (SCS), Davies-Bouldin Score (DBS), and Dunn's Index (DI) [55]. SCS measures the similarity of an object to its cluster (cohesion) compared to

Table 4: Comparison of the Performance of Different DCMs

Dataset	ML Models	Performance Metrics			
		Accuracy	Precision	Recall	F1-Score
Healthguard	NN	0.88	0.88	0.88	0.87
	DT	0.93	0.93	0.93	0.93
	LR	0.88	0.88	0.88	0.88
UQVS	NN	0.98	0.98	0.99	0.98
	DT	0.99	0.98	0.99	0.98
	LR	0.92	0.90	0.92	0.91
Diabetes	NN	0.66	0.65	0.66	0.65
	DT	0.79	0.78	0.79	0.78
	LR	0.75	0.75	0.75	0.75

Table 5: Comparison of the Performance of Different ADMs

Dataset	ML Models	Performance Metrics		
		DBS	SCS	DI
Healthguard	K-Means Clustering	2.57	0.18	0.057
	DBSCAN	0.681	0.732	0.235
UQVS	K-Means Clustering	1.841	0.098	0.072
	DBSCAN	0.517	0.612	0.412
Diabetes	K-Means Clustering	1.026	0.265	0.041
	DBSCAN	2.4	-.042	0.141

other clusters (separation). The high value of SCS (close to +1) of a cluster specifies its likeness with its clusters, whereas a low value (close to -1) indicates the opposite. In our experiment, we measure the SCS of clusters by taking the mean of all data points of that particular cluster, defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. DBS finds the average similarity measure of each cluster compared to its most similar cluster. The clustering algorithm having lower DBS shows better performance. Again, the DI value assesses the algorithm's compactness and cluster separation measure. Unlike SCS and DBS, a higher DI value implies better performance for clustering models [56]. It is apparent from the analysis that DBSCAN has outperformed K-means clustering based on our experimental setup.

We considered 3 different DCMs (i.e., NN, DT, and LR) and 2 different ADMs (i.e., DBSCAN, k-Means clustering) for our framework. These 3 different DCMs represent different ML models based on their complexity. The DT constraints are very straightforward and can be directly used in formal threat modeling. The LR models are comparatively complex models with a single nonlinear function. Since the nonlinear function is linearized with a Taylor series expansion, there is some loss of information. The 3rd of ML model (i.e., the NN model) is chosen since its constraints are computationally expensive to solve as it uses several nonlinear functions for prediction. For ADM, we consider two different types of clustering-based models. The choice of the models is motivated due to their high adoption in healthcare research. However, other models can be more suitable choices depending on the data type, the relationship among the features, and between the feature space and output class. We tabularize the pros and cons of several possible DCMs and ADMs in Table 6.

4.2. Feature Selection for ADM

As discussed before, our considered ADM contemplates the relationship between features (in this case, sensor measurements) to capture the alteration through adversarial attempts. However, we do not consider the relationship among all features for modeling the ADM. Finding relationships among all features creates a better ADM but overcomplicates the constraints, increasing solver complexity and thus making the threat analysis infeasible to solve in polynomial time. Besides, to draw a concave hull in a d-dimensional space, we need at least d-1 points. Many clusters violate this constraint while all feature relationships are considered together. As discussed in Section 4.1, we consider a DBSCAN-based ADM considering all feature relationships. The DBSCAN model is trained with optimal hyper-parameters. However, some outlier/ abnormal samples exist for the pair of features consideration model. Although the pair of features relationship model misses some abnormalities that can be found in all features together consideration model, our experimentation demonstrated in Table 4.1 shows that the pair of feature considerations can capture 95.43% and 96.64%

Table 6: Comparison between Different DCMs and ADMs

Type of ML Model	ML Model	Pros	Cons
DCM	DT [25]	Simple to understand and interpret, can capture non-linear relationships and doesn't require extensive data preprocessing like normalization, etc.	Prone to overfitting with deep trees, sensitive to minor data tweaks, and often less accurate compared to other methods.
	LR [26]	Simple and interpretable, it provides probabilities for outcomes and can be regularized to prevent overfitting.	Assumes a linear relationship between predictors and response and may miss complex data interactions or relationships.
	NN [27]	Can model complex, non-linear relationships and is effective in processing large, high-dimensional datasets and handling high-dimensional spaces.	Requires extensive data and computing power, lacks interpretability due to its black-box nature, and is sensitive to hyperparameters.
	Support Vector Machine (SVM) [57]	Is versatile due to different kernel functions, can handle non-linear boundaries, and is effective in high-dimensional spaces.	Unsuitable for vast datasets, requires feature scaling, and is sensitive to hyperparameters.
	k-Nearest Neighbors (kNN) [58]	Simple and intuitive. Can capture non-linear boundaries.	Prediction can be slow for large datasets. Sensitive to irrelevant features and the scale of the data. Need to choose the right number of neighbors (k).
	Naïve Bayes (NB) [59]	Works well with high-dimensional data and requires minimal training data to estimate parameters.	Makes a strong assumption on feature independence and may underperform if this assumption is breached.
	Random Forest (RF) [60]	Outperforms single trees, handles large, intricate datasets, and captures variable non-linearities and interactions.	Less interpretable than single trees, computationally demanding in training, and needs precise parameter tuning.
ADM	DBSCAN [29]	Detects varied cluster shapes, identifies outliers, and doesn't require specifying the number of clusters.	Underperforms with clusters of varying densities and is sensitive to distance metrics and hyperparameters.
	K-Means Clustering [28]	Processes large datasets quickly and efficiently and performs well when anomalies cluster separately from normal data points.	Assumes equally sized, spherical clusters, requires a predefined cluster count (k), and is sensitive to initial centroids.
	Isolation Forest (IF) [61]	Processes large datasets efficiently, is tailored for anomaly detection in high-dimensional data, and identifies global and local anomalies.	Its random nature can yield inconsistent results and is less interpretable than alternative methods.
	One-Class SVM [62]	Identifies non-linear boundaries between normal and anomalous data and works well with high-dimensional data.	Computationally expensive for large datasets and sensitive to hyperparameters.
	Autoencoder (AE) [63]	Captures non-linear and complex patterns and excels at detecting anomalies with unique features absent in normal data.	Needs deep learning resources, is computationally heavy, lacks interpretability, and may inaccurately reconstruct anomalies, making detection difficult.
	Principal Component Analysis (PCA) [64]	Reduces dimensionality for simpler visualization and analysis and can detect anomalies through reconstruction error analysis.	Presumes anomalies cause large reconstruction errors, and its linear approach might miss non-linear relationships.
	Local Outlier Factor [65]	Accounts for local density variations, aids anomaly detection in diverse datasets and doesn't need prior outlier counts.	Sensitive to hyperparameters and can be computationally intensive for large datasets.

Table 7: Performance Analysis of Pair of Relationship Model

Model	Healthguard Dataset		UQVS Dataset		Diabetes Dataset	
	# Benign Samples	# Outlier Sample	# Benign Samples	# Outlier Sample	# Benign Samples	# Outlier Sample
All Measurements Considered	13822	3178	182316	26799	731	37
Combination of Pair of Measurements Considered	13967	3033	183215	25900	736	32

anomalies for the synthetic and the UQVS dataset, respectively. Hence, the pair of features consideration model resembles the ADM developed with all features.

4.3. Adoption of Combined Model

As discussed before, we consider an SHS considering DT-based DCM and DBSCAN-based ADM. The DT-based model tends to find splitting points of the sensor measurements to clearly distinguish a group of vital sign measurements from one patient's status to another. However, DT does not consider the

Table 8: Attack Scenario under Different Attack Model

Type of Measurements	Heart Rate	Systolic	Diastolic	Blood Oxygen
Actual Measurements	122.94	75.98	153.56	93.2
Attacked Measurements (DT Constraints Only)	122.94	73.46	153.56	98.5
Attacked Measurements (both DT and DBSCAN Constraints)	120.332	81.855	149.67	98.5

interrelation between all other sensor values for a particular state, creating a need for an ADM and a DCM. Consequently, leveraging a clustering algorithm in the SHS model can accumulate the relationship between all sensor measurements for a particular patient state and make the system robust. Although NN-based DCMs might capture the inter-relationship between sensor measurements, clustering-based approaches are required for outlier detection as NN always puts a label, ignoring the possibility of data being an outlier. Considering such a model imposes constraints on sensor measurement alteration for the adversaries. An attacker can not alter a patient's status with only the knowledge of DT-based constraints. By compromising sensor values, the attacker could generate a sample satisfied by the DT-based model but labeled as an anomaly by the DBSCAN-based model, as demonstrated in the case studies. Accordingly, our proposed threat analyzer takes constraints from both DCM and ADM to acquire stealthy attack vectors.

5. Case Studies

We conduct two case studies to evaluate our framework. We consider both case studies an SHS with a DT-based DCM and a DBSCAN-based ADM (as per Section 4).

5.1. Case Study using a Synthetic Dataset

To verify SHChecker, we developed a realistic healthcare dataset with 8 sensor measurements with 17,000 samples this synthetic dataset, we have 6 labels of various patient statuses. Table 8 shows 4 sensor measurements of a particular patient. We employ our threat analysis framework to identify potential attack vectors to misclassify patients with high blood cholesterol to different patient statuses with minimal measurement alteration. Initially, the controller makes control decisions based on the standalone DCM (i.e., DT model). Accordingly, our framework found an attack vector that can misclassify the patient as a high blood pressure patient by manipulating two sensor measurements (i.e., systolic blood pressure and blood oxygen). Moreover, the framework also reported that no patient status alteration FDI attack goal could be successful if the attacker has accessibility to craft only a single measurement. Hence, the SHS with DT-based DCM is a 2-resilient model for any considered attack target. We further experiment with finding attack vectors from a more robust model that integrates a DBSCAN-based ADM and the DCM. Our threat analyzer successfully discovers an attack vector with minimal perturbation from such a robust model. However, the attack vector identified by the combined model requires altering more measurements than a standalone model's. For instance, to achieve the same goal of misclassifying the high blood cholesterol patient as a high blood pressure one, the attacker needs to alter all 4 measurements.

The proposed SHChecker framework reports the attack vector with minimal alteration from the combined model for the considered targeted attack, which requires decreasing the heart rate sensor measurement by 2.12%, increasing the systolic blood pressure measurement by 7.17%, decreasing the diastolic blood pressure measurement by 2.53%, and increasing the blood oxygen measurement value by 5.68%. The DT constraints and a portion of DBSCAN constraints extracted for the threat analysis have been demonstrated in Tables 9 and 10.

5.2. Case Study using a Real Dataset

We also verified our framework using two real-world datasets. We are showing a case study on a dataset collected by the University of Queensland [30]. The dataset contains 49 sensor measurements of 32 anesthesia patients with surgical cases at the Royal Adelaide Hospital, monitored using Philips Intellivue monitors and Datex-Ohmeda anesthesia machine. After removing the uncorrelated measurements with the labels, we considered 26 sensor measurements from the dataset for 209,115 samples. The monitoring systems issue

Table 9: Example DT Algorithm-Driven Constraints

$$inference(\mathcal{S},0) \rightarrow ((\mathcal{S}_5 \leq 20.5) \wedge (((\mathcal{S}_7 \leq 8.5) \wedge (\mathcal{S}_4 \leq 94.005)) \vee ((\mathcal{S}_7 \leq 8.5) \wedge (\mathcal{S}_2 \leq 140.46) \wedge (\mathcal{S}_0 \leq 100.51)))) \vee ((\mathcal{S}_5 > 20.5) \wedge (\mathcal{S}_3 > 130.495)))$$

$$inference(\mathcal{S},1) \rightarrow ((\mathcal{S}_5 > 20.5) \wedge (\mathcal{S}_3 \leq 130.495) \wedge (\mathcal{S}_2 \leq 140.46))$$

$$inference(\mathcal{S},2) \rightarrow ((\mathcal{S}_5 > 20.5) \wedge (\mathcal{S}_3 \leq 130.495) \wedge (\mathcal{S}_2 > 140.46))$$

$$inference(\mathcal{S},3) \rightarrow (\mathcal{S}_5 \leq 20.5) \wedge (\mathcal{S}_7 > 8.5) \wedge (\mathcal{S}_5 \leq 20.5)$$

$$inference(\mathcal{S},4) \rightarrow (\mathcal{S}_5 \leq 20.5) \wedge (\mathcal{S}_7 \leq 8.5) \wedge (\mathcal{S}_4 > 94.005) \wedge (\mathcal{S}_2 \leq 140.46)) \wedge (\mathcal{S}_0 > 100.51)$$

$$inference(\mathcal{S},5) \rightarrow (\mathcal{S}_5 \leq 20.5) \wedge (\mathcal{S}_7 \leq 8.5) \wedge (\mathcal{S}_4 > 94.005) \wedge (\mathcal{S}_2 > 140.46)$$

Table 10: Example DBSCAN Algorithm-Driven Constraints

$$\begin{aligned}
&...(((-0.75\mathcal{S}_{0,0,0}+0.35\mathcal{S}_{1,0,0}+79.929 \geq 0) \wedge (70.11 < \mathcal{S}_{1,0,0} \leq 70.86)) \oplus \\
&((-1.5\mathcal{S}_{0,0,1}+0.53\mathcal{S}_{1,0,1}+171.7767 \geq 0) \wedge (68.61 < \mathcal{S}_{1,0,1} \leq 70.11)) \oplus \\
&(((-0.08\mathcal{S}_{0,0,2}+0.53\mathcal{S}_{1,0,2}+171.7767 \geq 0) \wedge (65.21 < \mathcal{S}_{1,0,2} \leq 67.44)) \vee \\
&((0.08\mathcal{S}_{0,0,3}+0.39\mathcal{S}_{1,0,3}+1059.126 \geq 0) \wedge (85.27 < \mathcal{S}_{1,0,3} \leq 87.36))) \oplus \\
&((0.04\mathcal{S}_{0,0,4}0^0+0.43\mathcal{S}_{1,0,0}+21.929 \geq 0) \wedge (88.22 < \mathcal{S}_{1,0,4} \leq 92.33)) \oplus \\
&((0.04\mathcal{S}_{0,0,5}+0.41\mathcal{S}_{1,0,5}+21.21 \geq 0) \wedge (87.35 < \mathcal{S}_{0,0,5} \leq 91.15)))
\end{aligned}$$

single or multiple alarms based on the patient's vital signs. The dataset has a total of 58 labels having 28 different alarms. Out of these labels, 24 deal with single alarm, 19 handle a couple of alarms, 12 provide triple alarms, and the rest involve quadruple alarms. Using our threat analyzer, we identified several attack vectors that trigger wrong alarms, evading the DBSCAN and DT-based combined model. For instance, our experimentation shows that by altering 9.2%, 8.1%, 8.4%, 2.3%, 5.9%, 9.9%, 2.1%, 3.9% measurement alteration in artery diastolic pressure (ART Dia), artery mean pressure (ART Mean), effective end-tidal decreased hemoglobin oxygen saturation (ETDES) label, inspired decreased hemoglobin oxygen saturation (INDES) label, end-Tidal isoelectric (ETISO) point, inspired isoelectric (INISO) point, effective end-tidal concentration of sevoflurane (ETSEV) and inspired concentration of sevoflurane (INSEV) sensors respectively, an adversary can make the system trigger APNEA, low blood pressure (NBPsLOW), low end-tidal carbon-dioxide (etCO2LOW), high inspired concentration of sevoflurane (inSEVHIGH) alarm instead of APNEA, high minute volume (MVexpHIGH).

6. Evaluation

This section presents the findings from the considered SHS model and the feasibility of implementing our proposed framework. The ML models identified in Section 4 are considered for further evaluation. We present the threat analysis results of the SHS identified by SHChecker, considering the following research questions for evaluation.

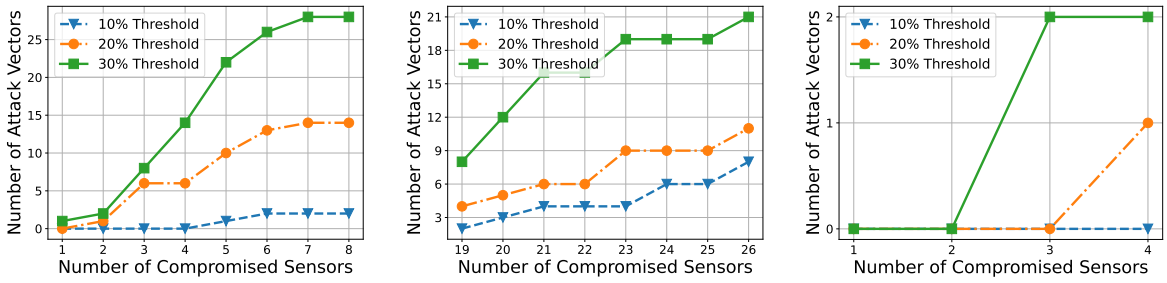
RQ1 What are the framework's findings in identifying attack vectors with variable attacker's capability? (Section 6.1.1)

RQ2 What are the most significant sensor measurements that will make the SHS most vulnerable while exploited by an FDI attack? (Section 6.1.2)

RQ3 What are the framework's findings about the resiliency of the SHSs? (Section 6.1.3)

RQ4 How feasible is implementing the proposed framework for a scalable CPS domain? (Section 6.2)

RQ5 Is the formalization correct? (Section 6.4)



(a) (b) (c)

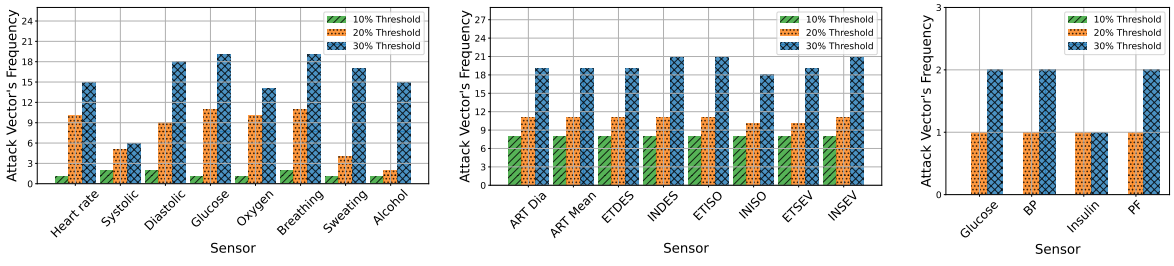
Figure 6: Performance analysis of the SHChecker framework for identifying attack vectors w.r.t. attacker's capability for (a) HealthGuard, (b) UQVS, and (c) Diabetes datasets.

6.1. Evaluation Results from SHS Threat Analysis

We evaluate the SHSs (i.e., with one synthetic and two real datasets) with SHChecker. The following three evaluation criteria have been considered.

6.1.1. Evaluation of Attack Vectors with Different Attacker's Capability

In this part, we evaluate the performance of our proposed framework by analyzing the total number of attack vectors concerning the attacker's capability. Figure 6 shows the number of found attack vectors for the different numbers of compromised sensors and the threshold for data injection in all 3 considered datasets. The figure shows from the HealthGuard dataset an attack vector can be found even when the attacker alters only one measurement. Three attack vectors can be found when the injected data is bounded within a threshold of 10% of the actual sensor measurements. The framework finds more attack vectors when the injection threshold is increased. The framework finds 28 different attack vectors when the attacker can manipulate eight sensor measurements, altering the measurement up to 30% of the actual value. Although SHChecker considers a combination of pairs of features, the attack vectors we get using formal threat analysis conform with the SHS ML models (i.e., attack vectors added with the sensor measurements are classified according to the attack goal by the DCM and verified by the ADM).



(a) (b) (c)

Figure 7: Frequency of the different sensor measurements in the attack vectors for (a) synthetic and (b) UQVS datasets.

6.1.2. Evaluation of System's Critical Sensor Measurements

SHChecker framework analyzes all the attack vectors and determines the participation of the individual sensor measurement in the vectors. We plot Figure 7 to represent the frequency of the sensor measurements in the attack vectors for all three datasets. From the Figure, we see that all the measurements, except a few (i.e., sensor measurement 2) in the HealthGuard dataset, participate in attack generation almost equally for the 30% attack threshold, which suggests all sensor measurements need to be protected with

Table 11: Devices to Compromise to Achieve an Attack Goal

Current State	Target State	# Sensors to Compromise	Sensors
Normal	High Cholesterol	3	Heart rate, Glucose, Alcohol
High Blood Pressure	Abnormal Oxygen Level	4	Systolic, Diastolic, Oxygen, Breathing
High Cholesterol	Excessive Sweating	2	Heart rate, Breathing

Table 12: Complexity Analysis of DCMs based on the Number of Sensor Measurements

Dataset	# Measurements	Number of Clauses		
		NN	DT	LR
Synthetic Data	8	2652	5196	2148
	10	2694	5104	2160
	12	2736	5349	2172
	14	2778	5465	2184
	16	2820	897	2196
UQVS Data	26	178962	56813	172608
	27	179005	56689	172666
	28	179048	57071	172724
	29	179091	57094	172782
	30	179134	59124	172840

equal importance if the attacker has 30% alteration capability. Moreover, from Figure 7(a), it is clear that the sweating and blood alcohol measurements influence less than the other measurements in the case of launching attacks. Hence, this study gives insight into which sensor measurements should get more attention while developing a defensive tool for the SHSs. If the frequency of a particular sensor measurement in attack vectors is much higher than the others, the sensors associated with the measurement must get extra attention to get secured. Thus, SHChecker can be an effective tool for providing an SHS design guide.

6.1.3. Evaluation of System's Resiliency

A system is said to be resilient to the degree to which it rapidly and effectively protects its critical capabilities from disruptions caused by adverse events and conditions. Our proposed threat model can determine the resiliency of a system for a particular attacker goal. Table 11 shows the resiliency table for the HealthGuard dataset, which conveys that an attacker cannot misclassify a normal patient into a high-cholesterol patient if he cannot access more than two devices. Hence, it can be inferred that the SHS is 2-resilient for that specific attack goal. Similarly, for the UQVS dataset, it appears that changing a patient label from normal to decrease in hemoglobin oxygen saturation (DESAT) label is 20-resilient, which signifies that an attacker can not achieve this attack goal by compromising 20 or fewer sensor measurements. The resiliency analysis capability of the framework provides a design guide specifying the relationship between the number of protected sensors and the risk reduction.

6.2. Scalability Analysis of SHChecker

We evaluate the SHChecker's scalability by analyzing the time requirement varying size of the SHS components (i.e., the number of sensor measurements). The model's scalability mainly depends on the time required to perform threat analysis for the solver based on the attacker's capability and number of sensor measurements, and this time is the most significant determinant of identifying critical attack vectors in a feasible time. Accordingly, we vary the number of SHS sensor measurements for analyzing the scalability of our system. Figures 8(a), 9(a), and 10(a) establish that execution time to create clusters from the DBSCAN constraints takes less time than the boundary creation time and the time increases linearly based on the number of sensor measurements. Figures 8(b), 9(b), and10(b) infers that the construction of DBSCAN constraints requires a lot more time than that of DT clusters. From Figures 8(c), 9(c), 10(c), 8(d), 9(d), and 10(d) it is apparent that increasing the attacker's capability increases the execution time for the solver as it requires more constraints to check. The growth rate of time required for the solver performing real-time threat analysis shows a rapid increment and raises scalability issues for large SHSs.

The complexity of the solver depends on the number of clauses. The time complexity is analyzed considering DT-based DCM and DBSCAN-based ADM. Table 12 demonstrates the number of clauses for various DCMs with varied numbers of sensor measurements. It is apparent from the table that DCMs have

limited dependency on the number of sensor measurements, and all three have almost a similar number of clauses. For the NN model, the number of clauses depends on the size of the system. We used a single-layer NN with 30 nodes in a hidden layer for the UQVS dataset. For the Healthguard dataset, the number of nodes in the hidden layer was 15, and we used 10 nodes in the Diabetes dataset. From Figure 11, we can observe that ADMs are mainly the reason for solver complexity, as increasing the number of features adds a lot of clauses to the solver. However, it is also clear from the figure that both of the ADMs in consideration have given rise to a similar number of clauses. As a result, scalability analysis of other DT and DBSCAN is sufficient to understand the time requirements for other models. Our experimentation found this value to be slightly more than 5 minutes for 30 sensor measurements. Hence, launching an attack on a patient monitored by an SHS whose sensor measurements do not vary drastically with time is possible.

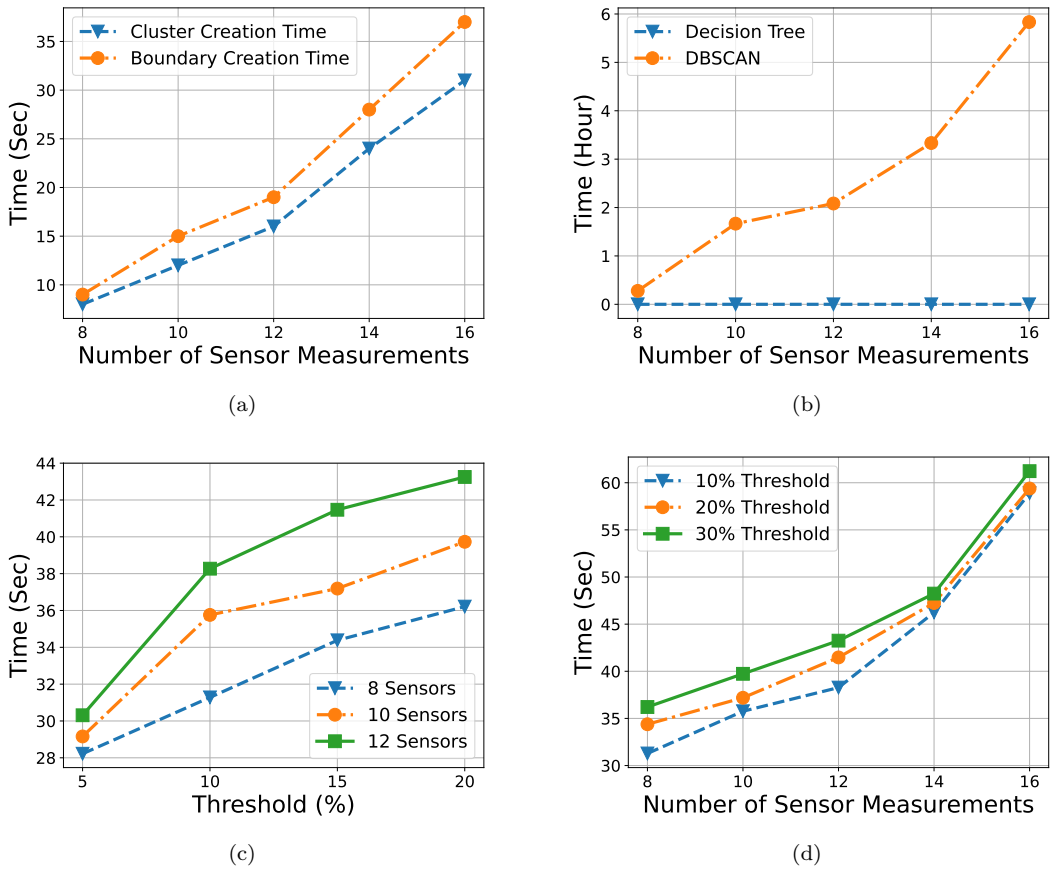


Figure 8: Execution time of the (a) cluster formation and boundary setup based on number of sensor measurements, (b) ML model's constraints extraction based on number of sensor measurements, (c) threat analysis based on threshold for data injection, and (d) threat analysis based on the number of sensor measurements measured from the HealthGuard dataset.

We divide the computation into 5 different phases. The corresponding complexity of each phase (i.e., time and space complexities) is included in Table 13. The threat vector identification is a formal threat synthesis approach using a satisfiability modulo theories (SMT)-based solver. A typical SMT solver (e.g., Z3) converts the model/input constraints into a set of logical/propositional clauses. The satisfiability of the clauses is assessed by optimal approaches like the Davis-Putnam-Logemann-Loveland (DPLL) algorithm, which uses backtracking and operates by choosing a variable, asserting a true or false value, and then simplifying the equations. The satisfaction of the simplified equations implies that the equations are also satisfied. If not, the checking process is recursively repeated using a different value for the variable. Since the DPLL algorithm splits the branches into binary conditions, the time complexity is $O(2^c)$, where c is

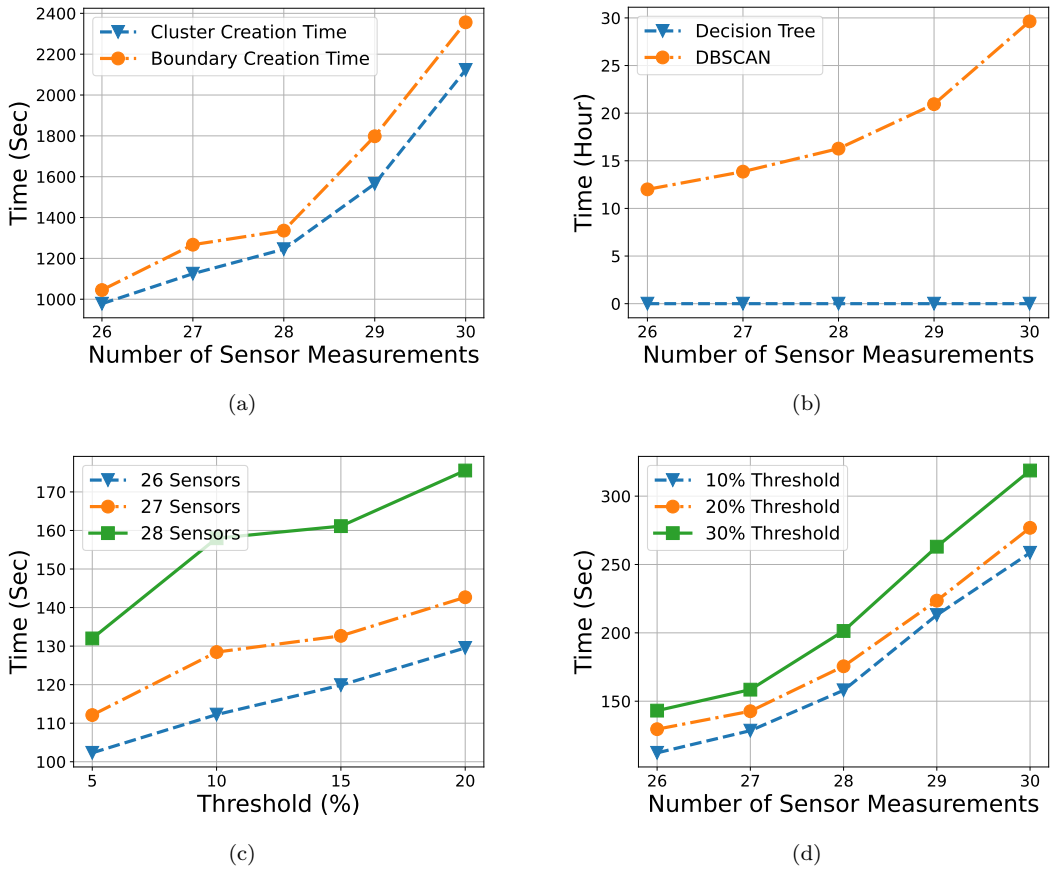


Figure 9: Execution time of the (a) cluster formation and boundary setup based on number of sensor measurements, (b) ML model's constraints extraction based on number of sensor measurements, (c) threat analysis based on threshold for data injection, and (d) threat analysis based on the number of sensor measurements measured from the UQVS dataset.

Table 13: Computation Complexity Analysis of SHChecker

Phase		**Time Complexity**	**Space Complexity**	**Notation Description**
ADMs' Cluster Formation	DBSCAN	$O(\|D\|^2)$	$O(\|D\|)$	D: All data samples $\|D\|$: Number of data samples for model training
	K-Means Clustering	$O(\|D\| \times d \times k \times I)$		d: Dimensionality of data k: Number of clusters I: Number of iteration for convergence
ADMs' boundary-setting		$O(\|D\|^2)$	$O(\|\mathcal{H}^E\| + \|\mathcal{H}^V\|)$	$\|\mathcal{H}^E\|$: Number of concave hul edges $\|\mathcal{H}^V\|$): Number of concave hull vertices
DCMs' constraints extraction	DT	$O(\|\mathcal{N}^{DT}\|)$	$O(\|\mathcal{N}^{NN}\|)$	$\|\mathcal{N}^{DT}\|$: Number of nodes of a DT model
	LR	$O(n_l)$	$O(n_l)$	n_l: Number of classes\labels
	NN	$O(\|\mathcal{N}^{NN}\|)$	$O(\|\mathcal{N}^{NN}\|)$	$\mathcal{N}^{NN}$: Number of nodes of an NN model
Attack Constraints Formulation		$O(\|\mathcal{S}\|)$	$O(\|\mathcal{S}\|)$	$\|\mathcal{S}\|$ Number of sensor measurements
Threat Vector Extraction		$O(2^c)$	$O(\|\mathcal{S}\|)$	c: Number of clauses

the number of unknown variables (in the constraints) to which the solver must assign values. To reduce the search cost and to find a more time-efficient solution, the DPLL algorithm employs a pruning strategy

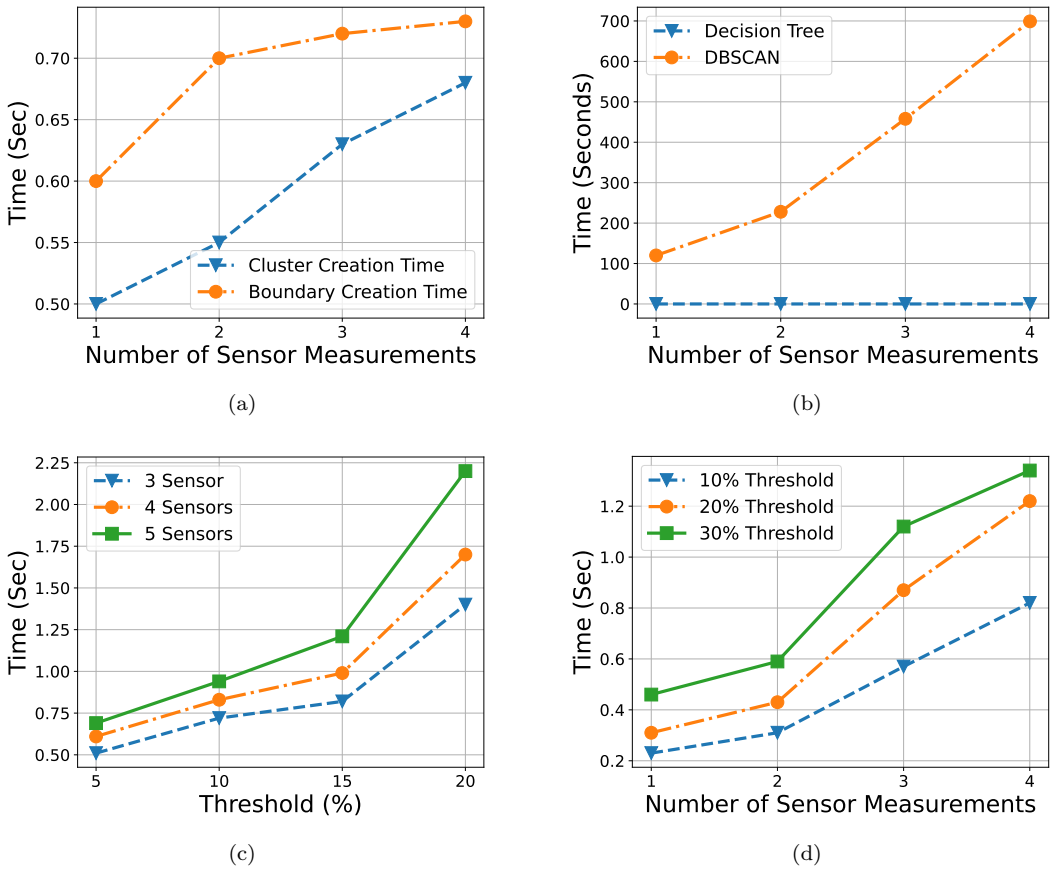


Figure 10: Execution time of the (a) cluster formation and boundary setup based on number of sensor measurements, (b) ML model's constraints extraction based on number of sensor measurements, (c) threat analysis based on threshold for data injection, and (d) threat analysis based on the number of sensor measurements measured from the Diabetes dataset.

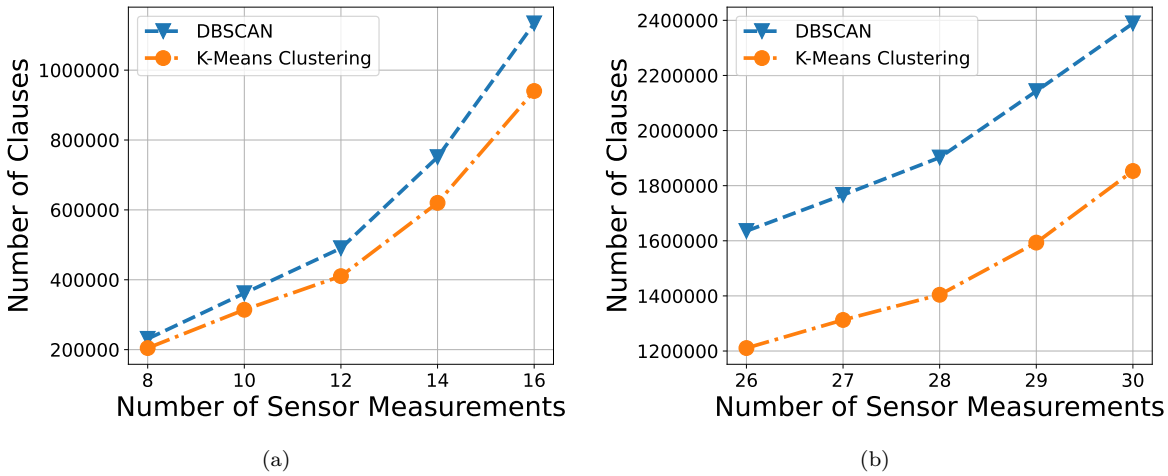


Figure 11: Complexity analysis of ADMs w.r.t. the number of sensor measurements for (a) synthetic and (b) UQVS datasets.

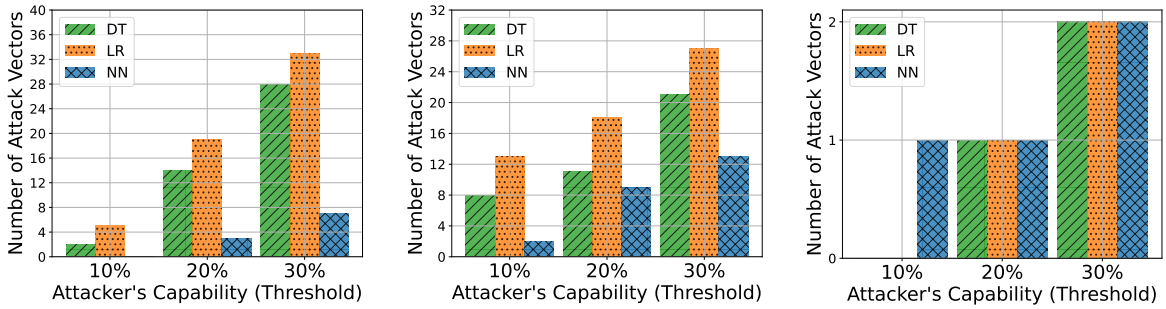


Figure 12: Threat analysis of various ML models for (a) synthetic and (b) UQVS datasets.

based on falsified/conflicting clauses to navigate the search space. Hence, the obtained cost is often much smaller than the worst-case $O(2^c)$ scenario. The overall time complexity of the proposed framework does not usually exceed $O((|D|)^2)$, while the space complexity lies within $O(|D|)$.

6.3. Threat Analysis of Various ML-based DCMs

Here, we evaluate the robustness of SHS based on the underlying DCM in terms of the attack vector count. We compare the associated attack vectors of various ML-based DCMs. Figure 12 shows the attack vector comparison of various DCMs (i.e., keeping DBSCAN as ADM) for both of our datasets in consideration. Our comprehensive analysis shows that LR-based DCM seems more vulnerable than the others. Besides, if the underlying DCM is NN instead of DT, the number of attack vectors appears to be fewer. Although the performance of the DCM of SHS with the NN-based model measured on performance metrics is slightly less than the DT-based model, the latter model is subjected to more threats. Therefore, choosing the NN model over DT yields a more robust model against FDI attacks. Hence, SHChecker can provide a design guide (i.e., choosing an ML model) for CPS design by enumerating the attack vectors associated with various ML models.

6.4. Formal Correctness Assessment

The authors concur on the importance of verifying the correctness of formalization. To evaluate the correctness of formalization, we formulated another research question (i.e., RQ5). We divide the correctness checking into two steps. First, we check the correctness of the DCMs, as shown in Table 14, and then the correctness of ADMs and attack modeling is assessed and demonstrated in Table 15. To examine the correctness of the DCM constraints, we first add only the DCM inference rules into the solver and check the model with all the dataset samples. We denote the solver's model as a formal model. The formal model's performance is compared with actual and approximate models. The approximate model uses the linearization approaches used to model the constraints of the formal model. The approximate model is a function like the actual model. On the other hand, the formal model is the output of a solver (i.e., modeled DCM inference in this case) after solving the solver constraints, provided the constraints are satisfiable. From Table 13, the F1-score of the approximate and formal models are the same, which verifies the formal correctness of the DCM constraints. DT performance is the same for all the models since the DT rules are directly used for the formal modeling without loss of information. However, the performance deviations for LR and NN models are due to the linearization carried out in the constraint modeling.

For analyzing the correctness of ADM and attack/attacker modeling, we create a function resembling the ADM constraints for the combination of a pair of measurements considered in the model. We check the function with all the attack vectors identified by SHChecker and find that the function identifies all the attack vectors as attacks. Thus, we can conclude the correctness of the formalization process. We can see that there is some discrepancy in the actual ADM. The reason is that the combination of the pair of measurements considered model loses some detail of the high dimensional feature space. However, the attack

Table 14: Checking the Formalization of DCMs

Dataset	ML Models	F1-Score		
		Actual Model	Approximate Model	Formal Model
Healthguard	NN	0.87	0.84	0.84
	DT	0.93	0.93	0.93
	LR	0.88	0.86	0.86
UQVS	NN	0.98	0.89	0.89
	DT	0.98	0.98	0.98
	LR	0.91	0.87	0.87
Diabetes	NN	0.65	0.64	0.64
	DT	0.78	0.78	0.78
	LR	0.75	0.72	0.72

Table 15: Checking the Formalization of ADM and Attacker's Property

Dataset	ML Models	Attack Threshold	Percentage of Successful Attacks	
			All Measurements Considered Model	Combination of pair of Measurement Considered Model
Healthguard	DBSCAN	10%	98.7%	100%
		20%	100%	100%
		30%	100%	100%
	K-Means Clustering	10%	99.4%	100%
		20%	100%	100%
		30%	100%	100%
UQVS	DBSCAN	10%	97.7%	100%
		20%	99.5%	100%
		30%	100%	100%
	K-Means Clustering	10%	97.1%	100%
		20%	99.2%	100%
		30%	100%	100%
Diabetes	DBSCAN	10%	98.5%	100%
		20%	100%	100%
		30%	100%	100%
	K-Means Clustering	10%	99.2%	100%
		20%	100%	100%
		30%	100%	100%

samples (i.e., samples found by injecting/adding attack vectors with the benign measurements) identified with high attack thresholds maintain large margins from the benign samples, and hence, those samples are identified as attacks by both models.

7. Related Work

Security is always a considerable concern while implementing a CPS. Popular cryptographic algorithms cannot be deployed in IoMT-based SHS sensors due to limited computation power, one of the most alarming problems. Some research attempts to introduce a new lightweight cryptographic algorithm that does not require massive computational power. Gong et al. propose a lightweight private homomorphism algorithm along with an encryption algorithm developed using the concept of DES that can be implemented in an SHS [66]. Sharma et al. propose a privacy preservation scheme for WBSN-based healthcare using multipath routing, secret sharing, and hashing [67]. However, the proposed approaches and solutions are lightweight yet vulnerable to several attacks. Hence, various security and threat analyses of IoT-enabled systems are getting attention. The existing security analysis of IoT-enabled CPSs can be broadly categorized into 3 types.

7.1. Regulation-based Security Analysis of IoT-enabled CPSs

Several works concern well-known regulations-based security analysis. Stellios et al. proposed a novel risk-based process for attack path identification and assessment against critical IoT-enabled systems leveraging customary building blocks such as Common Vulnerabilities and Exposures (CVE) and Common Vulnerability Scoring System (CVSS) [68]. Bakhshi et al. analyzed and categorized the industrial IoT-specific

threats from the data accumulation and abstraction layer of Cisco and Microsoft Azure cloud reference model [69]. Akatyev et al. assessed possible IoT-specific threats for futuristic smart homes enabled with several heterogeneous elements and advanced decision-making capabilities [70]. Casola et al. proposed an automated mechanism for threat modeling and risk assessment based on the threat catalog developed and enhanced in the context of the FP7 SPECS project of the affecting communication protocol and software components of IoT systems [71].

7.2. Rule-based Formal Security Analysis of IoT-enabled CPSs

Formal threat and resiliency analysis of rule-based IoT systems is well-experimented and explored [18, 19]. Mohsin et al. analyzed the network topology and interdependencies among the system components and developed a formal security analysis framework for the IoT-based systems, accordingly extracting potential attack vectors from integrity and availability types of attacks [18]. The proposed framework can extract potential attack vectors from integrity and availability types of attacks and determine the system's resiliency for an IoT-enabled CPS, considering the variable accessibility and capabilities of the attacker. The authors proposed another formal data-driven framework capable of machine-interpretable semantic modeling of IoT configurations in another work to identify security configuration abnormalities and IoT-specific attack vectors [19]. However, the proposed solutions are limited to analyzing the security of rule-based IoT systems. Unlike the existing formal threat analyzers, our proposed threat analysis framework can identify stealthy attack vectors from ML-based control systems with an integrated ML-based abnormality detection system.

7.3. Threat Analysis of ML-based IoT-enabled CPSs

Few works consider threat analysis using ML-based models. Luo et al. proposed an adversarial ML-based partial-model attack in the data fusion/aggregation process of IoT by only controlling a small part of the sensing devices [72]. Newaz et al. analyzed the threat space of ML-based SHS leveraging adversarial ML-based attacks with white-box and black-box settings. They figured out that the attack vectors can significantly degrade the performance of an ML-based SHS in detecting diseases and normal activities of the patients correctly, which eventually leads to erroneous treatment [73].

Acquiring and solving constraints from an ML-based system is way more challenging and different than that of rule-based systems. Formal analysis of Deep NN-based ML models has gained a lot of focus in several contemporary research works. Several efficient tools (e.g., Reluplex, Sherlock, Marabou, etc.) have been developed for verifying DNN [21, 22, 23]. Dreossi et al. attempted to identify issues in the case of applying a formal method in ML and analyze ML-based system behavior in the presence of environment uncertainty [74]. Verification of other ML algorithms using formal modeling has also been attempted. Törnblom et al. presented a tool called VoTE (Verifier of Tree Ensembles) for verifying DT-based ensemble techniques supporting up to 25 trees. Souri et al. formally verified a hybrid ML-based approach for fault prediction in IoT applications that incorporates Multi-layer Perceptron (MLP) and Particle Swarm Optimization (PSO) algorithms [75]. Unlike these research efforts, our proposed framework performs formal threat analysis of CPSs incorporating two different purpose ML-based models for attack synthesizing. It has opened a novel research direction in the ML-based formal modeling domain.

Other than the literature mentioned above, a few other works consider attack detection through network packet analysis using ML-based approaches [76]. However, attack detection is out of the scope of our work. Hinta et al. listed the vulnerabilities in one of the most critical IoT protocols (i.e., message queue telemetry transport) [77]. The proposed framework does not identify vulnerabilities in the IoT protocols. Instead, it identifies FDI attack vectors and possible attack impacts from an exploited IoT system. Tan et al. proposed a convolution neural network-based threat detection approach from artificial IoT-enable system [78]. Ramaki et al. proposed a framework to detect advanced persistent threats of critical infrastructure by graph-based attackers' behavior modeling [79]. The framework primarily attempts to perform correlation analysis of logged events. Later, an event correlation graph is generated from those events, followed by extracting attack paths on attack detection. However, these approaches can only identify attacks from learned threat patterns. The proposed framework, on the other hand, formally synthesizes a verifiable attack path from an ML-based CPS.

8. Limitations and Future Works

Formalization is the main scope of our work. Since ML models are opaque, formalization efforts attempt to extract constraints from such models and analyze them to provide the same flexibility as transparent physics-based models. Our main contribution is to formalize a realistic CPS modeled with the complex interaction of a supervised learning-based DCM and unsupervised learning-based ADM. The constraints extraction from ML models was a challenging and novel effort. We further formalize a hazardous attack (i.e., false data injection) and attacker, which allows us to extract, interpret, and analyze provable threat space of safety-critical CPSs. The formality of our framework is a major advantage, which enables a human user/decision maker to observe the vulnerable region of an intricated defense-ware CPS model. However, SHChecker provides several features that augment the advantages other than formality. For instance, the proposed framework can analyze the resiliency and scalability of ML-based CPSs. However, the SHChecker framework requires some modification for complete automation and scalability. This section discusses our work's limitations and future extension plans.

8.1. SHChecker Limitations

The significant limitations in the manuscript are as follows.

1. The framework's main shortcoming is limited scalability, so it cannot identify attack vectors for significantly large models in a feasible time frame.
2. We need to linearize the underlying ML models in this framework, and the approximation models may lose some essential details of the model, which prevents identifying some attack vectors.
3. SHChecker, in its current form, is not applicable for identifying threats from time-series anomaly detection models
4. The framework also cannot identify threats from multi-label disease datasets.

An alternative non-formal approach, specifically the RL-based threat analysis, often exhibits superior computational efficiency compared to formal methods.

1. The RL-based techniques can outperform the formal methods regarding computational efficiency.
2. The RL-based approaches can work with the actual ML models to identify attack vectors, although identifying the attack vector is not guaranteed.
3. RL-based approaches can outperform the formal ones when the problem space is too large for the latter to handle. Additionally, when solution time is constrained, RL-based approaches can provide suboptimal solutions, whereas formal solvers may not provide any solution.

8.2. Future Extension Plan

We have identified the limitations of our proposed framework to plan for the future extension. We will focus on the following plans for future extension of our work.

1. The framework will be extended to work with multi-label disease datasets. Moreover, we will develop a formal IoMT system model with multi-sensor fusion and analyze the threat space of that system, exploring the added challenges.
2. We will develop a systematic method for determining the attack threshold in the system since the threshold is dependent on the domain.
3. The updated version of the framework will automate the linearization process for ML models' constraint extraction.

4. Currently, the ML models' constraint extraction supports a limited set of DCMs and ADMs. In our upcoming works, we will provide threat analytics solutions for all existing supervised and unsupervised ML models. We will further explore identifying attack vector extraction from ensembled-based ML models.

5. Advanced persistent threats can stay in the system for a long time and stealthily attack multiple times to launch a successful attack. We will factor those threats into our upcoming work.

6. Our future work aims to create formal model constraints considering all feature relations since the pair of feature considerations in our ADM produces some false-negative outcomes.

7. The work will be further extended with an automated CPS reconfiguration scheme leveraging the attack vectors identified by the proposed framework.

9. Conclusion

This paper presents a formal threat analysis framework to model and study the security of ML-based CPS. The tool can analyze the potential threats that satisfy the attacker's goal. We consider IoMT-enabled SHS as a case study for such a system that uses ML to understand the relationships between the sensor measurements and the consistency among the measures. We exploit this knowledge to perform formal analysis to synthesize potential attack vectors for a given attack model, where the attacker can change the patient's health status (the actual one) to the wrong one (the targeted state). Our evaluation results on synthetic and real datasets show that stealthy FDI attacks can be launched in distinct ways by compromising different numbers and types of sensor measurements, even compromising only one sensor measurement in some cases.

Acknowledgement

This research was partially supported by National Science Foundation (NSF) under Award #1929183.

References

[1] P. Jayalaxmi, R. Saha, G. Kumar, M. Alazab, M. Conti, X. Cheng, Pignus: A deep learning model for ids in industrial internet-of-things, Computers & Security (2023) 103315.
[2] A. C. B. Monteiro, R. P. França, R. Arthur, Y. Iano, An overview of the internet of medical things (iomt): Applications, benefits, and challenges, Security and Privacy Issues in Internet of Medical Things (2023) 83–98.
[3] S. Chauhan, N. Arora, R. Arora, Iot and machine learning-based smart healthcare system for monitoring patients, in: Artificial Intelligence of Health-Enabled Spaces, CRC Press, 2023, pp. 1–19.
[4] E. Lawrence, Nearly half of americans delayed medical care due to pandemic, https://khn.org/news/nearly-half-of-americans-delayed-medical-care-due-to-pandemic/, accessed: 2021-04-21 (2020).
[5] N. Jiwani, K. Gupta, P. Whig, Machine learning approaches for analysis in smart healthcare informatics, in: Machine Learning and Artificial Intelligence in Healthcare Systems, CRC Press, 2023, pp. 129–154.
[6] B. M. Reddy, Amalgamation of internet of things and machine learning for smart healthcare applications–a review, Int. J Comp. Eng. Sci. Res 5 (2023) 08–36.
[7] I. Y. Tyukin, D. J. Higham, A. N. Gorban, On adversarial examples and stealth attacks in artificial intelligence systems, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–6.
[8] L. Zhang, K. Sridhar, M. Liu, P. Lu, X. Chen, F. Kong, O. Sokolsky, I. Lee, Real-time data-predictive attack-recovery for complex cyber-physical systems, in: 2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS), IEEE, 2023, pp. 209–222.
[9] A. I. Newaz, A. K. Sikder, M. A. Rahman, A. S. Uluagac, A survey on security and privacy issues in modern healthcare systems: Attacks and defenses, ACM Transactions on Computing for Healthcare 2 (3) (2021) 1–44.
[10] Iot thermostat bug allows hackers to turn up the heat, https://blog.newskysecurity.com/iot-thermostat-bug-allows-hackers-to-turn-up-the-heat-948e554e5e8b, accessed: 2023-08-29 (2017).
[11] A. Sears, Felt so violated: Milwaukee couple warns hackers are outsmarting smart homes, https://www.fox6now.com/news/felt-so-violated-milwaukee-couple-warns-hackers-are-outsmarting-smart-homes, accessed: 2023-08-29 (2019).
[12] A. B. Wang, 'i'm in your baby's room': A hacker took over a baby monitor and broadcast threats, parents say, https://www.washingtonpost.com/technology/2018/12/20/nest-cam-baby-monitor-hacked-kidnap-threat-came-device-parents-say/, accessed: 2023-08-29 (2018).

[13] K. Paul, Dozens sue amazon's ring after camera hack leads to threats and racial slurs, https://www.theguardian.com/technology/2020/dec/23/amazon-ring-camera-hack-lawsuit-threats, accessed: 2023-08-29 (2020).
[14] S. Morgan, Patient insecurity: Explosion of the internet of medical things, https://cybersecurityventures.com/patient-insecurity-explosion-of-the-internet-of-medical-things/, accessed: 2022-05-10 (2019).
[15] A. Barua, Y. G. Achamyeleh, M. A. A. Faruque, A wolf in sheep's clothing: Spreading deadly pathogens under the disguise of popular music, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 277–291.
[16] M. A. Rahman, P. Bera, E. Al-Shaer, Smartanalyzer: A noninvasive security threat analyzer for ami smart grid, in: 2012 Proceedings IEEE INFOCOM, IEEE, 2012, pp. 2255–2263.
[17] M. A. Rahman, A. Jakaria, E. Al-Shaer, Formal analysis for dependable supervisory control and data acquisition in smart grids, in: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2016, pp. 263–274.
[18] M. Mohsin, Z. Anwar, G. Husari, E. Al-Shaer, M. A. Rahman, Iotsat: A formal framework for security analysis of the internet of things (iot), in: 2016 IEEE conference on communications and network security (CNS), IEEE, 2016, pp. 180–188.
[19] M. Mohsin, Z. Anwar, F. Zaman, E. Al-Shaer, Iotchecker: A data-driven framework for security analytics of internet of things configurations, Computers & Security 70 (2017) 199–223.
[20] N. I. Haque, M. A. Rahman, D. Chen, H. Kholidy, Biota: Control-aware attack analytics for building internet of things, in: 2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), IEEE, 2021, pp. 1–9.
[21] G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochenderfer, Reluplex: An efficient smt solver for verifying deep neural networks, in: International Conference on Computer Aided Verification, Springer, 2017, pp. 97–117.
[22] S. Dutta, S. Jha, S. Sanakaranarayanan, A. Tiwari, Output range analysis for deep neural networks, arXiv preprint arXiv:1709.09130.
[23] G. Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al., The marabou framework for verification and analysis of deep neural networks, in: International Conference on Computer Aided Verification, Springer, 2019, pp. 443–452.
[24] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, M. A. Amanullah, I. Abaker Targio Hashem, E. Ahmed, M. Imran, Clustering-based real-time anomaly detection—a breakthrough in big data technologies, Transactions on Emerging Telecommunications Technologies (2019) e3647.
[25] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1986) 81–106.
[26] D. W. Hosmer Jr, S. Lemeshow, R. X. Sturdivant, Applied logistic regression, Vol. 398, John Wiley & Sons, 2013.
[27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.
[28] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, Oakland, CA, USA, 1967, pp. 281–297.
[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: kdd, Vol. 96, 1996, pp. 226–231.
[30] D. Liu, M. Görges, S. A. Jenkins, University of queensland vital signs dataset: development of an accessible repository of anesthesia patient monitoring data for research, Anesthesia & Analgesia 114 (3) (2012) 584–589.
[31] J. W. Smith, J. E. Everhart, W. Dickson, W. C. Knowler, R. S. Johannes, Using the adap learning algorithm to forecast the onset of diabetes mellitus, in: Proceedings of the annual symposium on computer application in medical care, American Medical Informatics Association, 1988, p. 261.
[32] Healthguard-a-machine-learning-based-security-framework-for-smart-healthcare-systems, https://github.com/iqtidar27/HealthGuard-A-Machine-Learning-Based-Security-Framework-for-Smart-Healthcare-Systems/tree/main/Dataset (2019).
[33] N. I. Haque, M. Ngouen, M. A. Rahman, S. Uluagac, L. Njilla, Shatter: Control and defense-aware attack analytics for activity-driven smart home systems, arXiv preprint arXiv:2305.09669.
[34] Shchecker, https://github.com/imtiazulhaque/research-implementations/tree/main/shchecker (2020).
[35] R. Kyusakov, J. Eliasson, J. Delsing, J. Van Deventer, J. Gustafsson, Integration of wireless sensor and actuator nodes with it infrastructure using service-oriented architecture, IEEE Transactions on industrial informatics 9 (1) (2012) 43–51.
[36] K. Wang, X. Bai, J. Li, C. Ding, A service-based framework for pharmacogenomics data integration, Enterprise Information Systems 4 (3) (2010) 225–245.
[37] N. Pereira, B. Andersson, E. Tovar, Widom: A dominance protocol for wireless medium access, IEEE Transactions on Industrial Informatics 3 (2) (2007) 120–130.
[38] M. Hoskins, Reviewing the new all-inclusive dario glucose meterAccessed: 2020-01-09.
[39] B. L. Sng, D. J. Tan, C. W. Tan, N.-L. R. Han, R. Sultana, A. T. H. Sia, A preliminary assessment of vital-signs-integrated patient-assisted intravenous opioid analgesia (vpia) for postsurgical pain, BMC anesthesiology 20 (1) (2020) 1–8.
[40] Hypertension, https://catalog.data.gov/dataset/hypertension/ (April 2018).
[41] R. Martin, R. Ratan, M. Reding, T. Olsen, Higher blood glucose within the normal range is associated with more severe strokes, Stroke research and treatment.
[42] A. S. Bhogal, A. R. Mani, Pattern analysis of oxygen saturation variability in healthy individuals: Entropy of pulse oximetry signals carries information about mean oxygen saturation, Frontiers in physiology 8 (2017) 555.
[43] M. A. Pimentel, A. E. Johnson, P. H. Charlton, D. Birrenkott, P. J. Watkinson, L. Tarassenko, D. A. Clifton, Toward a robust estimation of respiratory rate from pulse oximeters, IEEE Transactions on Biomedical Engineering 64 (8) (2017)

1914–1923.
[44] J. Fell, R. Voas, The effectiveness of a 0.05 blood alcohol concentration (bac) limit for driving in the united s tates, Addiction 109 (6) (2014) 869–874.
[45] J. Wiens, E. S. Shenoy, Machine learning for healthcare: on the verge of a major shift in healthcare epidemiology, Clinical Infectious Diseases 66 (1) (2018) 149–153.
[46] H. Kaur, S. K. Wasan, Empirical study on applications of data mining techniques in healthcare, Journal of Computer science 2 (2) (2006) 194–200.
[47] K. Srinivas, B. K. Rani, A. Govrdhan, Applications of data mining techniques in healthcare and prediction of heart attacks, International Journal on Computer Science and Engineering (IJCSE) 2 (02) (2010) 250–255.
[48] A. Moreira, M. Santos, Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points.
[49] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, Knn model-based approach in classification, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2003, pp. 986–996.
[50] D. Storm, Medjack: Hackers hijacking medical devices to create backdoors in hospital networks, `https://www.computerworld.com/article/2932371/medjack-hackers-hijacking-medical-devices-to-create-backdoors\protect\@normalcr\relax-in-hospital-networks.html`, accessed: 2020-01-08 (2015).
[51] V. Pournaghshband, M. Sarrafzadeh, P. Reiher, Securing legacy mobile medical devices, in: International Conference on Wireless Mobile Communication and Healthcare, Springer, 2012, pp. 163–172.
[52] L. Liu, X. Xu, Y. Liu, Z. Ma, J. Peng, A detection framework against cpma attack based on trust evaluation and machine learning in iot network, IEEE Internet of Things Journal 8 (20) (2021) 15249–15258.
[53] W. Ding, H. Hu, L. Cheng, Iotsafe: Enforcing safety and security policy with real iot physical interaction discovery, in: the 28th Network and Distributed System Security Symposium (NDSS 2021), 2021.
[54] C. Goutte, E. Gaussier, A probabilistic interpretation of precision, recall and f-score, with implication for evaluation, in: European conference on information retrieval, Springer, 2005, pp. 345–359.
[55] Z. Halim, J. H. Khattak, Density-based clustering of big probabilistic graphs, Evolving systems 10 (3) (2019) 333–350.
[56] Cluster validation essentials, `https://www.datanovia.com/en/lessons/cluster-validation-statistics-must-know-\protect\@normalcr\relaxmethods/`, accessed: 2020-04-22 (2019).
[57] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[58] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.
[59] R. O. Duda, P. E. Hart, et al., Pattern classification and scene analysis, Vol. 3, Wiley New York, 1973.
[60] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.
[61] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 eighth ieee international conference on data mining, IEEE, 2008, pp. 413–422.
[62] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, Neural computation 13 (7) (2001) 1443–1471.
[63] G. E. Hinton, R. Zemel, Autoencoders, minimum description length and helmholtz free energy, Advances in neural information processing systems 6.
[64] H. Hotelling, Analysis of a complex of statistical variables into principal components., Journal of educational psychology 24 (6) (1933) 417.
[65] M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.
[66] T. Gong, H. Huang, P. Li, K. Zhang, H. Jiang, A medical healthcare system for privacy protection based on iot, in: 2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), 2015, pp. 217–222.
[67] N. Sharma, R. Bhatt, Privacy preservation in wsn for healthcare application, Procedia computer science 132 (2018) 1243–1252.
[68] I. Stellios, P. Kotzanikolaou, C. Grigoriadis, Assessing iot enabled cyber-physical attack paths against critical systems, Computers & Security 107 (2021) 102316.
[69] Z. Bakhshi, A. Balador, J. Mustafa, Industrial iot security threats and concerns by considering cisco and microsoft iot reference models, in: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), IEEE, 2018, pp. 173–178.
[70] N. Akatyev, J. I. James, Evidence identification in iot networks based on threat assessment, Future Generation Computer Systems 93 (2019) 814–821.
[71] V. Casola, A. De Benedictis, M. Rak, U. Villano, Toward the automation of threat modeling and risk assessment in iot systems, Internet of Things 7 (2019) 100056.
[72] Z. Luo, S. Zhao, Z. Lu, Y. E. Sagduyu, J. Xu, Adversarial machine learning based partial-model attack in iot, in: Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning, 2020, pp. 13–18.
[73] A. I. Newaz, N. I. Haque, A. K. Sikder, M. A. Rahman, A. S. Uluagac, Adversarial attacks to machine learning-based smart healthcare systems, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–6.
[74] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, S. A. Seshia, Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems, in: International Conference on Computer Aided Verification, Springer, 2019, pp. 432–442.
[75] A. Souri, A. S. Mohammed, M. Y. Potrus, M. Malik, F. Safara, M. Hosseinzadeh, Formal verification of a hybrid machine learning-based fault prediction model in internet of things applications, IEEE Access 8 (2020) 23863–23874.
[76] R. Rawat, V. Mahor, B. Garg, M. Chouhan, K. Pachlasiya, S. Telang, Modeling of cyber threat analysis and vulnerability in iot-based healthcare systems during covid, in: Lessons from COVID-19, Elsevier, 2022, pp. 405–425.

[77] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, S. Karuppayah, Mqtt vulnerabilities, attack vectors and solutions in the internet of things (iot), IETE Journal of Research (2021) 1–30.
[78] L. Tan, K. Yu, F. Ming, X. Cheng, G. Srivastava, Secure and resilient artificial intelligence of things: a honeynet approach for threat detection and situational awareness, IEEE Consumer Electronics Magazine 11 (3) (2021) 69–78.
[79] A. A. Ramaki, A. Ghaemi-Bafghi, A. Rasoolzadegan, Captain: Community-based advanced persistent threat analysis in it networks, International Journal of Critical Infrastructure Protection 42 (2023) 100620.