SAM: Foreseeing Inference-Time False Data Injection Attacks on ML-enabled Medical Devices

Mohammadreza Hallajiyan
The University of British Columbia
Vancouver, Canada
hallaj@ece.ubc.ca

Athish Pranav Dharmalingam
Indian Institute of Technology Madras
Chennai, India
cs21b011@smail.iitm.ac.in

Gargi Mitra
The University of British Columbia
Vancouver, Canada
gargi@ece.ubc.ca

Homa Alemzadeh
University of Virginia
Charlottesville, USA
ha4d@virginia.edu

Shahrear Iqbal
National Research Council
Fredericton, Canada
shahrear.iqbal@nrc-cnrc.gc.ca

Karthik Pattabiraman
The University of British Columbia
Vancouver, Canada
karthikp@ece.ubc.ca

Abstract

The increasing use of machine learning (ML) in medical systems necessitates robust security measures to mitigate potential threats. Current research often overlooks the risk of adversaries injecting false inputs through peripheral devices at inference time, leading to mispredictions in patients' conditions. These risks are hard to foresee and mitigate during the design phase since the system is assembled by end users at the time of use. To address this gap, we introduce SAM, a technique that enables security analysts to perform System Theoretic Process Analysis for Security (STPA-Sec) on ML-enabled medical devices during the design phase. SAM models the medical system as a control structure, with the ML engine as the controller and peripheral devices as potential points for false data injection. It interfaces with state-of-the-art vulnerability databases and Large Language Models (LLMs) to automate the discovery of vulnerabilities and generate a list of possible attack paths. We demonstrate the usefulness of SAM through case studies on two FDA-cleared medical devices: a blood glucose management system and a bone mineral density measurement software. SAM allows security analysts to expedite the security assessment of ML-enabled medical devices at the design phase. This proactive approach mitigates potential patient harm and reduces costs associated with post-deployment security measures.

CCS Concepts

• **Security and privacy** → **Systems security**;

Keywords

ML-enabled Medical Devices, False Data Injection, STPA-Sec, Security Assessment

ACM Reference Format:
Mohammadreza Hallajiyan, Athish Pranav Dharmalingam, Gargi Mitra, Homa Alemzadeh, Shahrear Iqbal, and Karthik Pattabiraman. 2024. SAM: Foreseeing Inference-Time False Data Injection Attacks on ML-enabled Medical Devices. In *Proceedings of the 2024 Workshop on Cybersecurity in Healthcare (HealthSec '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3689942.3694745

1 Introduction

There is a growing trend to incorporate machine learning (ML) into medical applications [31] to automate remote patient monitoring, enhance diagnosis and treatment, and improve healthcare accessibility in remote locations with limited availability of medical experts. Over 800 such devices are registered with the U.S. FDA as of today [35] with some used in life-critical scenarios, such as live surgical assistance [38] and generation of treatment plans [39]. Therefore, mispredictions by the ML algorithms may lead to health complications and even death of the patient. Common causes of such mispredictions are issues with the training data, device failures, and security attacks [18, 41]. This paper focuses on security attacks, *specifically inference-time false (input) data injection attacks*, on ML-enabled medical systems at the time of use.

Detecting and preventing false data injection attacks in ML-enabled medical systems is challenging. Malicious inputs are difficult to detect due to the complexity and often unexplainable nature of ML models [3], along with variations in patients' physiological characteristics. Therefore, it is imperative to prevent such attacks. However, preventing these attacks in ML-enabled medical systems is equally challenging because of the large and complex attack surface, involving multiple interconnected devices. At the core of an ML-enabled medical system is an ML-enabled device[1] that acts as a controller (e.g., a blood glucose monitoring app), processes inputs from peripheral sensor devices (e.g., glucose meters, smartwatches), makes predictions/decisions (e.g., insulin doses) and sends outputs to actuators (e.g., insulin pumps). Unlike other ML-enabled systems (such as industrial systems and automotive), the final assembly of medical systems is done by end users (patients or healthcare providers) oblivious to security risks. Moreover, they might have multiple options for each peripheral device (e.g., glucose meters from various brands), each having different vulnerabilities. Even if the ML-enabled software is secure, the system is at risk if vulnerable peripherals are connected, allowing adversaries to inject malicious inputs. It is difficult for developers to foresee these security risks.

---

[1] This ML-enabled device is either software that can be installed on a general-purpose computer or one that is bundled with proprietary hardware.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the national government of Canada. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Request permissions from owner/author(s).

HealthSec '24, October 14–18, 2024, Salt Lake City, UT, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1238-8/24/10
https://doi.org/10.1145/3689942.3694745

Existing efforts to secure ML-enabled medical systems focus on enhancing device security and ML model resilience [24]. However, even resilient ML models can be vulnerable to new attacks [5]. Evaluating the ML model's security alone fails to address scenarios where adversaries inject malicious inputs through peripheral devices and communication channels. Anticipating these scenarios would enable device manufacturers[2] as well as independent security analysts to foresee and mitigate post-deployment security risks early, enhancing the safety of critical medical systems. This would reduce costs associated with post-attack damage control and accelerate security assessments through automation, minimizing human errors. It would also facilitate the evaluation of system security whenever new vulnerabilities in peripheral devices are reported.

To enable this analysis, we developed **SAM** (STPA-Sec for ML-enabled Medical Devices), a technique to identify system-wide security risks due to false data injection in ML-enabled medical systems. SAM adopts system-theoretic process analysis for security (STPA-Sec) [43], a technique for identifying how an adversary can exploit security vulnerabilities in interconnected components to cause the system to malfunction, potentially leading to a safety hazard. Along the same principles and assuming the ML model is running on a secure cloud-based environment, SAM aims to detect *all possible ways* an adversary can exploit known vulnerabilities in peripheral devices and communication channels to inject malicious data into the ML-enabled device, leading to either misdiagnoses or incorrect treatment plans, eventually harming the patients. The process involves four steps, namely (1) modeling the medical system as a control structure, (2) understanding the ML technique's susceptibility to malicious data, (3) identifying insecure control actions that could cause mispredictions, and (4) identifying all possible malicious data injection paths leading to these insecure actions.

Designing SAM presents two main challenges. **First,** an ML-enabled medical system includes multiple peripheral devices, and a user has multiple choices for each device, each of which can have vulnerabilities in various technological layers (e.g., communication protocols, third-party software, operating systems, firmware) or due to human interactions and operating environments. Identifying all these vulnerabilities is challenging. To address this, we abstract these devices into technological layers and operational factors, subsequently identifying known vulnerabilities in these layers using databases like MITRE CVE [17] and the National Vulnerability Database [19]. **Second,** understanding how each vulnerability can cause the ML model to mispredict and affect patients requires expertise in ML vulnerabilities, system security, and the medical domain. To facilitate this, we use LLMs trained on extensive medical and attack information. We design LLM queries (prompts) to automate the identification of potential attack paths for data injection and assess the impact of mispredictions on patients, using the medical system's description and the vulnerabilities in its peripherals.

Contributions: Our contributions are summarized as follows.

- We propose a technique for comprehensively identifying vulnerabilities in the ML techniques, and vulnerabilities in peripheral devices of ML-enabled medical systems.
- We devise an LLM-powered technique to automate the identification of potential false data injection paths, their impact on ML model predictions, and the consequences for patients.
- We perform two case studies using SAM on - (1) d-Nav [37], a blood glucose monitoring system and, (2) ABMD [34], a bone mineral density calculator. These studies demonstrate that SAM can identify potential attack paths, their impact on ML predictions, and the resulting health hazards for patients.

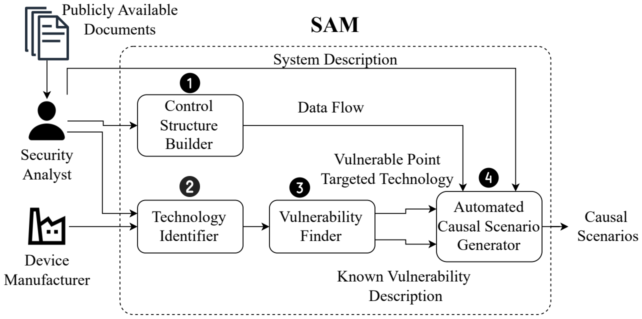


Figure 1: Overall Workflow of SAM

2 Approach: SAM

We propose SAM, a technique for systematically conducting STPA-Sec on AI/ML-enabled medical devices. Figure 1 provides a high-level overview of SAM. SAM consists of four components as follows:
(1) Control structure Builder guides the user (device manufacturer/security analyst) in building the control structure of the ML-enabled device under assessment, which describes the system components, their communications, and data flow paths (details in §2.1);
(2) Technology Identifier outputs the ML technique used in the system and all the technological and operational factors associated with each peripheral device (§2.2);
(3) Vulnerability Finder outputs the known vulnerabilities for each technological factor and finds out the latest reports on data injection attacks on ML technique used by the system (§2.3);
(4) Automated Causal Scenario Generator uses an LLM with a meticulously crafted prompt, based on information about identified technologies, ML techniques, and known vulnerabilities, to generate causal scenarios. These scenarios outline all possible ways an adversary can inject malicious data points into the system and the potential impacts on patient safety (§2.4).

2.1 Control Structure Builder

To perform STPA-Sec, SAM first requires the control structure of the system under assessment. The control structure gives an end-to-end view of the system components, their interconnections, and data flow paths within the system, which will be used to find the insecure control actions that might result in system hazards and to identify potential attack entry points and paths.

Different medical systems have different control structures, requiring users to derive a new control structure for each system when performing STPA-Sec. While manufacturers can refer to the system design, third-party security analysts must rely on publicly disclosed device descriptions, which are often incomplete and lack

---

[2] A security expert working in the same company and having the same knowledge about the ML-enabled device as its manufacturer

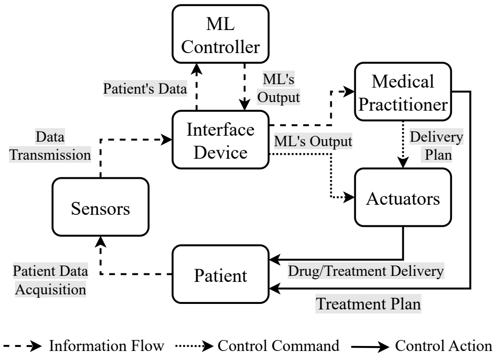


Figure 2: Generalized Control Structure for ML-enabled Medical Devices

standardization, making the process time-consuming and error-prone. To mitigate this, MITRE [15] recommends developing a standardized method for describing components and their interactions. The control structure builder offers a template to help users easily and accurately derive any medical system's control structure.

We incorporate a generalized control structure in SAM that covers all necessary components and interconnections in an ML-enabled medical system, as shown in Figure 2. It comprises six primary elements: the *Patient, Sensor, Actuator, Medical Practitioner, Interface Device*, and *ML Controller*. The *ML Controller* is either a Software-as-Medical-Device (SaMD) installable on general-purpose devices or a Software-in-Medical-Device (SiMD) integrated with proprietary hardware. Methods employed for data exchange among the different components include manual data entry and physical or wireless technologies such as Wi-Fi, Bluetooth, or Ethernet connections. The directed edges in the figure indicate these communication links and data flow paths. We created this generalized structure by manually examining the descriptions of 20 FDA-cleared ML-enabled medical devices, obtained from their FDA pre-approval summaries and manufacturers' websites. The security analyst can customize it (e.g., remove or add more instances of some components) to match the description of the system under assessment. For instance, there is no actuator in ECG devices (e.g., the CardioLogs Platform[36]) and hence, this must be removed. Instead, treatment is done by medical practitioners based on the controller output.

2.2 Technology Identifier

Our prior investigations [11] showed that ML techniques in medical systems are vulnerable to false data injection attacks, and technologies used in peripheral devices often serve as potential attack points. To enable a thorough security assessment of the technologies used in the system, the Technology Identifier component helps the user gather information on the ML technique used in the system and the technologies used in the peripherals.

2.2.1 Identifying the ML technique. When using SAM, device manufacturers can directly input the ML technique information. For independent security analysts, identifying the ML technique is challenging without the manufacturer's cooperation. In such cases, the Technology Identifier estimates the ML technique using publicly available information, such as FDA device descriptions and details from online sources such as the Medical Futurist website [32]. SAM utilizes a Natural Language Processing (NLP) technique (distilbert-base-uncased-distilled-squad [27]), paired with a set of targeted questions to extract information about device functionality (e.g., CT scan analysis), ML technique (e.g., deep learning technique), and input data type (e.g., DICOM files) from these sources. However, the NLP responses must be reviewed and validated manually.

Category	Factors
Human Interaction	• Data Entry / Supervision • Data Validation / authentication • Anomaly/failure detection
Communication Protocol	• Communication protocol and specific version • Data encryption
Electromagnetic Susceptibility	• Susceptibility to electromagnetic radiation • Repercussions of radiation exceeding threshold • Presence of shielding/mitigation strategy
Dependencies	• Firmware • Hardware • OS • External libraries

Table 1: Factors considered for security assessment

2.2.2 Identifying the peripheral device technologies. We have integrated two questionnaires into SAM. The first contains questions regarding the compatibility conditions for each peripheral device in the control structure and must be filled out by the manufacturer of the ML-enabled device. When SAM is used by an independent analyst, the compatibility information (e.g., compatible communication links, input device, operating system, etc.) can be inferred from the publicly available product description. For instance, for the glucose meter in a blood-glucose management system (BGMS), the compatibility condition might be Bluetooth connectivity. Similarly, certain BGMS apps might be compatible only with iOS devices.

Next, users must identify the technologies used in each compatible device. For this purpose, we integrate a second questionnaire into SAM covering a range of technological and operational factors. These questions ensure thorough coverage of potential attack entry points in the subsequent assessment steps. We have categorized the factors into four groups, namely *(i) Human Interaction, (ii) Communication Protocol, (iii) Electromagnetic Susceptibility*, and *(iv) Dependencies*, as outlined in Table 1. Our choice of factors is based on known attack vectors targeting ML-enabled medical devices [42].

Note that the SAM user must manually identify all commercial peripheral devices that meet the compatibility conditions specified by the ML-enabled device manufacturer. To complete the second questionnaire, the user can refer to FDA pre-market device summaries [40] and publicly available information on each peripheral, such as product descriptions on the manufacturer's website[3].

2.3 Vulnerability Finder

Next, SAM identifies all the known security vulnerabilities related to the identified technologies using the following process.

2.3.1 Retrieval of Potential ML Attacks. The Vulnerability Finder component of SAM identifies known ML attacks on the assessed device by interfacing with Google Scholar to retrieve relevant peer-reviewed articles. Because crafting effective search keywords is challenging, SAM uses a local LLM (Llama3 on Ollama [22]) to

---

[3] The questionnaires can be found at https://bit.ly/3XRorV6

generate precise keywords based on the device's functionality, ML technique, and input data. A customized web scraper then uses these keywords to find articles on inference-time attacks targeting similar ML techniques. We validated the results manually.

2.3.2 Retrieval of Known Vulnerabilities Using CVE Database. Once the SAM user has answered all the questions about the technological and operational factors in the questionnaires (see §2.2.2), SAM uses these responses as search keywords to find known vulnerabilities in the MITRE Common Vulnerability Enumeration (CVE) database [17]. By utilizing this constantly updated database, we ensure that SAM has captured all the known vulnerabilities linked to each technology. In the subsequent steps, SAM uses insights into vulnerable points, relevant technologies, and known vulnerabilities to automatically generate causal scenarios for the given hazards.

2.4 Automated Causal Scenario Generator

The most crucial step of the STPA-Sec process is identifying causal scenarios for predefined hazards within the system [43]. Performing this step manually makes it error-prone and time-consuming, as the user must rely on their knowledge of security and domain expertise. Furthermore, the process of finding a relationship between security vulnerabilities in peripherals, attack strategy, and its impact on the patient is complex. To address these problems, we leverage the capabilities of LLMs to automatically identify these causal scenarios based on the latest vulnerabilities found in the technologies used in the system under assessment. Note that we assume the hazards and unsafe control actions have already been identified using the STPA-Sec process and based on the security analyst's understanding of the system. Automating this step is beyond the scope of this work.

A key challenge when using LLMs is crafting the right prompt to obtain the ideal response for a given task. For SAM, an ideal response would include a detailed set of attack steps exploiting a peripheral vulnerability to perform inference-time data injection on a given ML technique. For this, we developed the following prompt.

"Act as a security engineer who has the task of identifying the steps that an adversary follows to cause a security breach in an ML-enabled medical system. <*Description of an ML-enabled medical system*>. <*Definition of security breach*>. You are given a system description, an ML attack, a targeted input peripheral component, and a known vulnerability in the input component. Give a list of steps to show how an adversary can exploit the vulnerability to mislead the ML-enabled component and how that affects the action of the output device on the patient.
System Description: <*The SAM user manually writes this description by inspecting information disclosed by the manufacturer.*>
Data flow: <*This can be derived from the control structure constructed using the Control structure builder in §2.1.*>
ML attack: <*The ML attack identified in §2.3*>
Targeted input peripheral component: <*One of the peripheral input devices in the control structure built in §2.1*>
Targeted technology: <*One of the underlying technologies in the input device, as identified by the technology identifier (§2.2)*>
Known vulnerability: <*Description of the known vulnerability in the targeted technology, as retrieved from the CVE database in §2.3*>"

We observed that explicitly assigning the LLM the role of a security analyst before giving it additional information improves the readability and relevance of the generated results - this is in line with other work in this area [29, 14, 28]. Similarly, mentioning the data flow provides clarity to the LLM regarding the sequence of data transmission between different components in the system.

By running this prompt for each vulnerable point in the system and each vulnerability uncovered at that point, SAM, regardless of the existence of safety/security margins, generates a comprehensive set of steps an adversary might take to compromise the security of an ML-enabled medical device. Device manufacturers or security analysts can then disregard those that have already been mitigated and develop design recommendations for the remaining ones.

3 Evaluation Methodology

3.1 Experimental Setup

3.1.1 Case Studies. We applied SAM to the following two FDA-cleared ML-enabled medical devices. We selected these mainly due to their extensive use by a substantial number of patients.

The d-nav system [37] is an ML-enabled app by Hygieia that prescribes insulin doses to Diabetic patients based on their past and present blood glucose levels and other physiological data such as meal and sleep timings. Subsequently, the patient or a caregiver reads the recommended insulin dose and configures the insulin pump to administer the specified amount to the patient's body.

The ABMD system [34] is an ML-enabled image processing software by HeartLung.AI that measures bone mineral density from CT scans. Subsequently, it generates a report for the medical practitioner that allows them to make treatment plans.

3.1.2 Implementation. We used Python to develop all components of SAM, except for the Control Structure Builder. Furthermore, in the *Automated Causal Scenario Generator*, we employed and evaluated three state-of-the-art large language models, viz., ChatGPT 4.0, ChatGPT 4o, and Llama 3. These models were selected for their advanced natural language processing capabilities and well-known effectiveness in generating contextually relevant outputs.

3.2 Research Questions

Our main research questions and the corresponding evaluation metrics used to answer them are as follows:

- **RQ1:** *How does the technology identifier facilitate the identification of all underlying technologies within an AI/ML-enabled medical device?* – In evaluating our technology identifier shown in §2.2, we emphasize its dual benefit: accelerating the identification process of ML techniques used in medical devices and enhancing the comprehensive identification of technologies across all peripheral devices to the best of our ability.
- **RQ2:** *How effective is the vulnerability identifier in helping security analysts identify ML attacks and known vulnerabilities specific to AI/ML-enabled medical devices?* – We evaluate the efficiency of the vulnerability identifier shown in §2.3 by demonstrating its ability to identify vulnerabilities reported in the underlying technology and their relevance to the system under assessment.

- **RQ3:** *How accurate, realistic, and detailed are the causal scenarios generated by LLMs?* – To evaluate the quality of the causal scenarios generated by the automated causal scenario generator (§2.4), we focus on three key aspects: (1) the correctness of the steps, measured as the fraction of completely correct steps among those generated, (2) the detail provided on the attacker's capabilities or actions, and (3) the inclusion of additional useful information, such as potential patient impact.

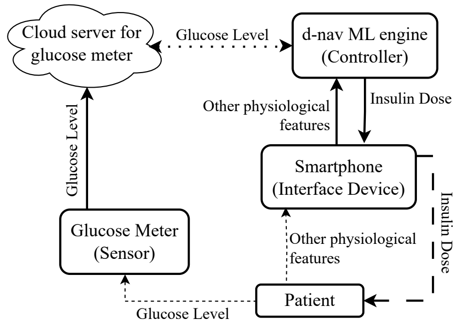
4 Experimental Results

We first build the control structures of the d-Nav and ABMD systems using the method proposed in §2.1. For identifying the number of instances of each component in the control structure and the presence of any additional device in the system, we referred to the device summaries submitted by the manufacturers for the FDA pre-market approval [37, 34] and their websites [10, 13].

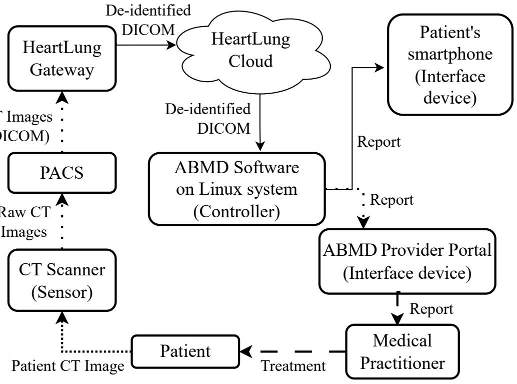
Figure 3a shows the control structure of the d-Nav system. The d-Nav app, installed on a smartphone, interfaces with an ML-enabled back end on a cloud server. Blood glucose values are entered manually or collected from a connected glucose meter via cloud-to-cloud integration. Based on the glucose values and other manually entered physiological data, the ML engine recommends an insulin dose, which is displayed on the app for the patient to adjust their insulin pump. Communication between the ML engine and smartphone occurs via the Internet. The directed edges indicate information flow and user actions. Figure 3b depicts the control structure of the ABMD system. The ML-enabled software runs on a Linux system. The patient's images captured by the CT scanner are transferred to a local PACS server via Wi-Fi, converted to DICOM format, and sent to a gateway server for deanonymization. The deanonymized files are then sent to the HeartLung cloud server, from where the ABMD software retrieves and processes them. The resulting bone mineral density report is displayed to a medical practitioner through a provider portal and sent to the patient's smartphone app.

Efficiency in Technology Identification (RQ1). The answer to RQ1 has two parts – identification of the ML technique and the identification of technological and operational factors. While unfortunately, for both systems (d-Nav and ABMD) the Technology Identifier could not retrieve the exact ML technique used due to a lack of sufficient public information, it was able to retrieve the device functionalities and input formats. It inferred that d-Nav is a software device that calculates the next dose of insulin based on blood glucose data trends captured by an automated blood glucose meter. As for the ABMD system, the Technology Identifier inferred that it is a software module that estimates bone mineral density in vertebral bones from input CT scans provided in DICOM format. We manually inspected the device documentation and found that the information that SAM retrieved matched ours, but manual inspection took much more time (approximately two hours) and effort as compared to that taken by SAM (a few seconds).

SAM identified six technology factors for d-Nav and nine for ABMD that can act as potential attack points, as depicted in Table 2 based on the factors shown in Table 1.


Cloud server for glucose meter
Glucose Level
d-nav ML engine (Controller)
Other physiological features
Insulin Dose
Glucose Level
Smartphone (Interface Device)
Insulin Dose
Glucose Meter (Sensor)
Other physiological features
Patient
Glucose Level

(a) Control Structure of the d-Nav System



(b) Control Structure of the ABMD system

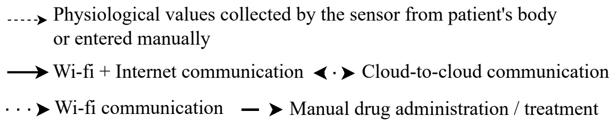

Physiological values collected by the sensor from patient's body or entered manually
Wi-fi + Internet communication
Cloud-to-cloud communication
Wi-fi communication
Manual drug administration / treatment

Figure 3: Control structures of the d-nav and ABMD systems constructed using the generalized control structure

Ability to discover vulnerabilities (RQ2). The answer to RQ2 has two parts: identification of inference-time attacks in the ML technique and identification of vulnerabilities in the underlying technologies of the peripheral devices. SAM could identify at least one highly relevant paper for both systems - a model inversion attack [12] for the d-Nav system, and a GAN-based malicious image tampering attack [16] for the ABMD system.

For the peripheral devices and communication protocols, SAM successfully retrieved all relevant CVE records. For instance, in the d-Nav system, SAM identified 593 CVE records [7] for the Wi-Fi communication between the glucose meter and the cloud server. For the ABMD software, SAM identified 541 CVE records [6] for the Ethernet communication between the CT scan device and the PACS. We observed that most of the recent CVE records for these technologies are either pending further analysis or lack an available workaround or patch.

The performance of LLMs (RQ3). Using the device description, and vulnerability information extracted in the previous steps, we crafted prompts for the LLM-based causal scenario generator using the template shown in § 2.4. For a given system, the user should

Device	Points of Vulnerability
d-Nav system	• Glucose meter • Wi-Fi communication between glucose meter and the cloud server • the communication between the cloud server for glucose meter and the ML controller • Wi-Fi communication between the interface device and the ML controller • interaction of interface with SQL database • Android OS on the interface
ABMD software	• Ethernet communication between the CT Scan Device to PACS • PACS Server • Ethernet communication between the PACS and Gateway • Gateway • Wi-Fi communication between the gateway and the ML controller • Wi-Fi communication between the patient's interface device and the ML controller • Wi-Fi communication between the provider's interface device and the ML controller • Linux OS on the provider's interface device • Android OS on the patient's interface device

Table 2: Identified points of vulnerability in d-Nav and ABMD

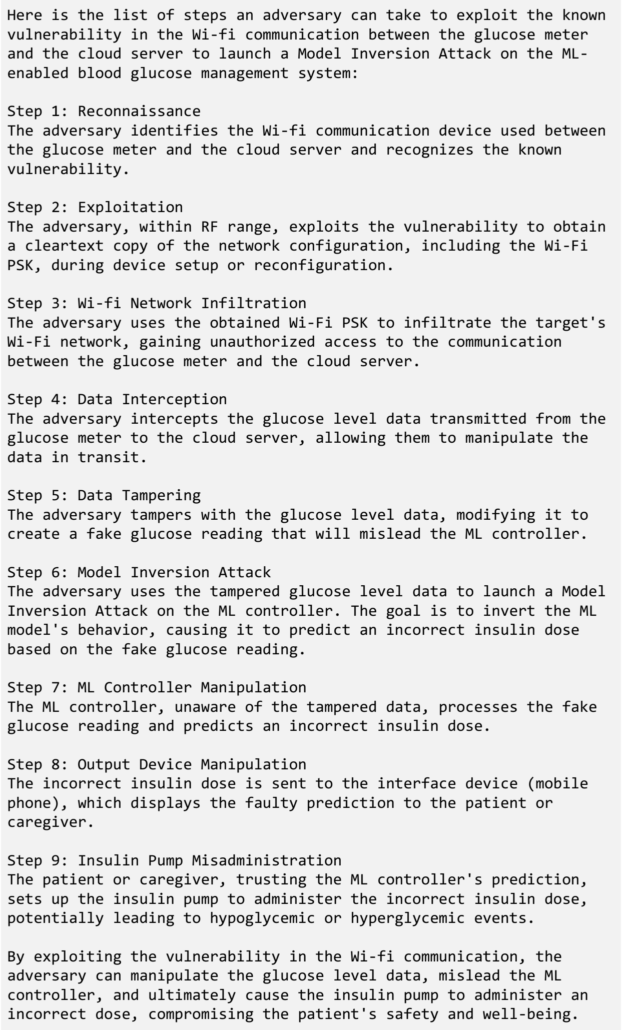
execute the prompt separately for every {hazard, ML attack, targeted peripheral component, peripheral device technology, vulnerability} tuple to get a comprehensive set of possible security attacks. We show the results for one such tuple for each medical system.

For d-Nav, we selected hypoglycemia as the hazard, and *"injecting excess insulin"* as the control action that causes it. We consider an adversary who conducts a model inversion attack (identified in the previous step) on the ML engine followed by false data injection. This attack makes the ML engine mispredict the insulin dose. To execute this attack, the adversary injects false glucose readings into the Wi-Fi channel that transmits the patient's glucose readings from the glucose meter to the ML engine running in the cloud server. We assume that the patient uses a Wi-Fi router with an unpatched known vulnerability, *CVE-2023-35836* [9], that the adversary exploits for injecting the malicious glucose readings. Figure 4 shows the response of the Llama3 LLM to the prompt.

For ABMD, we selected incorrect treatment as the hazard, and *"The medical practitioner not providing the correct diagnosis on the potential bone cracks"* as the control action that causes it. We consider an adversary who conducts a GAN-based malicious image tampering attack (identified in the previous step) on the ML engine. This attack makes the ML engine mispredict the bone mineral density. To execute this attack, the adversary tampers with the CT images by compromising the Ethernet communication between the CT scan device and the PACS. We assume that the patient uses an Ethernet driver with a known vulnerability, *CVE-2023-1670* [8], that the adversary exploits for tampering the CT images. Figure 5 shows the response of GPT4o to the prompt.

In both cases, the LLMs generated steps outlining how an adversary would execute the attacks. We evaluated responses as follows:

(1) **Correctness:** We manually validated and computed step-wise correctness of the LLM responses and observed the following: 92% (ChatGPT 4.0), 96% (ChatGPT 4o), and 90% (Llama 3) for d-Nav, and 90% (ChatGPT 4.0), 92% (ChatGPT 4o), and 84% (Llama 3) for ABMD. This demonstrates the effectiveness of different LLMs with our proposed prompts.



```
Here is the list of steps an adversary can take to exploit the known
vulnerability in the Wi-fi communication between the glucose meter
and the cloud server to launch a Model Inversion Attack on the ML-
enabled blood glucose management system:

Step 1: Reconnaissance
The adversary identifies the Wi-fi communication device used between
the glucose meter and the cloud server and recognizes the known
vulnerability.

Step 2: Exploitation
The adversary, within RF range, exploits the vulnerability to obtain
a cleartext copy of the network configuration, including the Wi-Fi
PSK, during device setup or reconfiguration.

Step 3: Wi-fi Network Infiltration
The adversary uses the obtained Wi-Fi PSK to infiltrate the target's
Wi-Fi network, gaining unauthorized access to the communication
between the glucose meter and the cloud server.

Step 4: Data Interception
The adversary intercepts the glucose level data transmitted from the
glucose meter to the cloud server, allowing them to manipulate the
data in transit.

Step 5: Data Tampering
The adversary tampers with the glucose level data, modifying it to
create a fake glucose reading that will mislead the ML controller.

Step 6: Model Inversion Attack
The adversary uses the tampered glucose level data to launch a Model
Inversion Attack on the ML controller. The goal is to invert the ML
model's behavior, causing it to predict an incorrect insulin dose
based on the fake glucose reading.

Step 7: ML Controller Manipulation
The ML controller, unaware of the tampered data, processes the fake
glucose reading and predicts an incorrect insulin dose.

Step 8: Output Device Manipulation
The incorrect insulin dose is sent to the interface device (mobile
phone), which displays the faulty prediction to the patient or
caregiver.

Step 9: Insulin Pump Misadministration
The patient or caregiver, trusting the ML controller's prediction,
sets up the insulin pump to administer the incorrect insulin dose,
potentially leading to hypoglycemic or hyperglycemic events.

By exploiting the vulnerability in the Wi-fi communication, the
adversary can manipulate the glucose level data, mislead the ML
controller, and ultimately cause the insulin pump to administer an
incorrect dose, compromising the patient's safety and well-being.
```

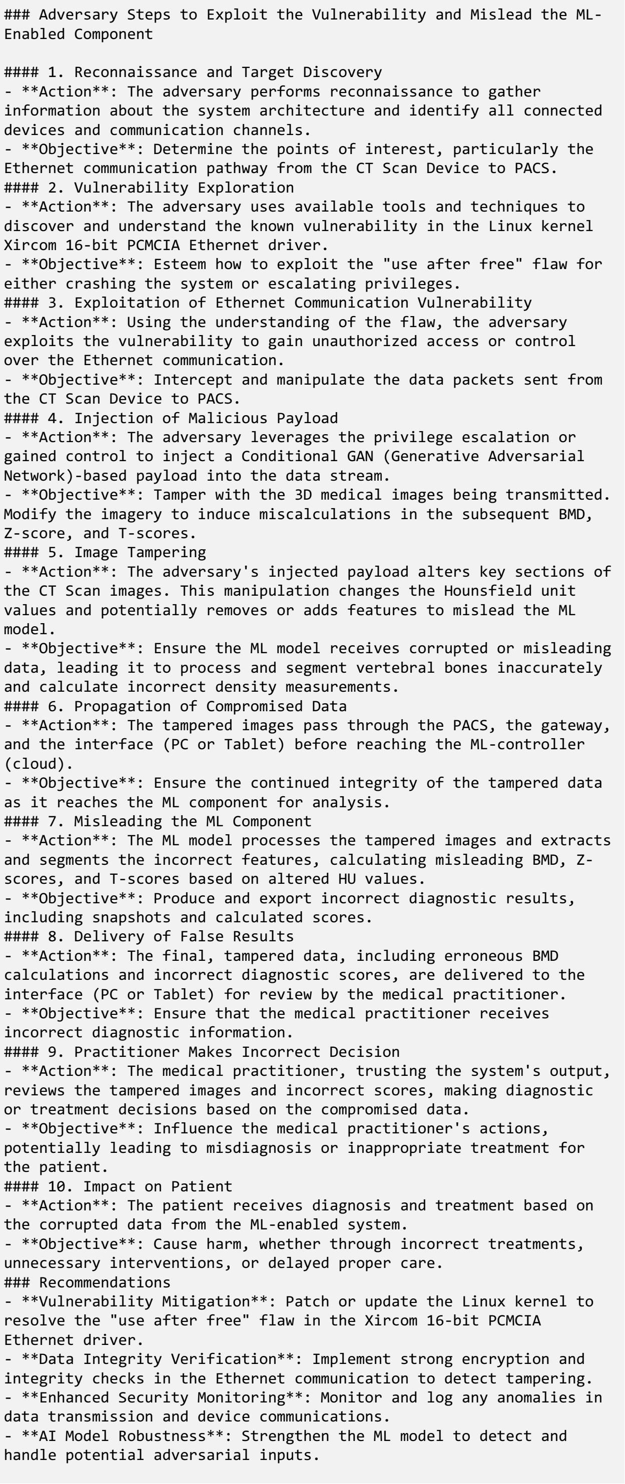
Figure 4: Generated causal scenario using Llama 3 for the Wi-Fi communication in d-Nav system

(2) **Level of Detail:** In both case studies, ChatGPT 4o provided more detailed steps compared to the other models. For most points of vulnerability, it clearly defined the objective of each step and the required actions to be taken. Llama 3 provided more details than ChatGPT 4.0 in some instances.

(3) **Additional Information:** In both cases, we found that ChatGPT 4.0 excelled in detailing attacks' impact on Confidentiality, Integrity, and Availability, and recommending mitigation strategies. It also summarized steps to exploit vulnerabilities. Llama 3 and ChatGPT 4.0 followed closely behind in terms of providing information on the impact of vulnerabilities on patients.

5 Related Work

There have been several tools that facilitate conducting STPA and STPA-Sec in different application domains. To determine their suitability for STPA-Sec on ML-enabled medical systems, we focus on three main features of the tools – whether they address security



```
### Adversary Steps to Exploit the Vulnerability and Mislead the ML-
Enabled Component

#### 1. Reconnaissance and Target Discovery
- **Action**: The adversary performs reconnaissance to gather
information about the system architecture and identify all connected
devices and communication channels.
- **Objective**: Determine the points of interest, particularly the
Ethernet communication pathway from the CT Scan Device to PACS.
#### 2. Vulnerability Exploration
- **Action**: The adversary uses available tools and techniques to
discover and understand the known vulnerability in the Linux kernel
Xircom 16-bit PCMCIA Ethernet driver.
- **Objective**: Esteem how to exploit the "use after free" flaw for
either crashing the system or escalating privileges.
#### 3. Exploitation of Ethernet Communication Vulnerability
- **Action**: Using the understanding of the flaw, the adversary
exploits the vulnerability to gain unauthorized access or control
over the Ethernet communication.
- **Objective**: Intercept and manipulate the data packets sent from
the CT Scan Device to PACS.
#### 4. Injection of Malicious Payload
- **Action**: The adversary leverages the privilege escalation or
gained control to inject a Conditional GAN (Generative Adversarial
Network)-based payload into the data stream.
- **Objective**: Tamper with the 3D medical images being transmitted.
Modify the imagery to induce miscalculations in the subsequent BMD,
Z-score, and T-scores.
#### 5. Image Tampering
- **Action**: The adversary's injected payload alters key sections of
the CT Scan images. This manipulation changes the Hounsfield unit
values and potentially removes or adds features to mislead the ML
model.
- **Objective**: Ensure the ML model receives corrupted or misleading
data, leading it to process and segment vertebral bones inaccurately
and calculate incorrect density measurements.
#### 6. Propagation of Compromised Data
- **Action**: The tampered images pass through the PACS, the gateway,
and the interface (PC or Tablet) before reaching the ML-controller
(cloud).
- **Objective**: Ensure the continued integrity of the tampered data
as it reaches the ML component for analysis.
#### 7. Misleading the ML Component
- **Action**: The ML model processes the tampered images and extracts
and segments the incorrect features, calculating misleading BMD, Z-
scores, and T-scores based on altered HU values.
- **Objective**: Produce and export incorrect diagnostic results,
including snapshots and calculated scores.
#### 8. Delivery of False Results
- **Action**: The final, tampered data, including erroneous BMD
calculations and incorrect diagnostic scores, are delivered to the
interface (PC or Tablet) for review by the medical practitioner.
- **Objective**: Ensure that the medical practitioner receives
incorrect diagnostic information.
#### 9. Practitioner Makes Incorrect Decision
- **Action**: The medical practitioner, trusting the system's output,
reviews the tampered images and incorrect scores, making diagnostic
or treatment decisions based on the compromised data.
- **Objective**: Influence the medical practitioner's actions,
potentially leading to misdiagnosis or inappropriate treatment for
the patient.
#### 10. Impact on Patient
- **Action**: The patient receives diagnosis and treatment based on
the corrupted data from the ML-enabled system.
- **Objective**: Cause harm, whether through incorrect treatments,
unnecessary interventions, or delayed proper care.
### Recommendations
- **Vulnerability Mitigation**: Patch or update the Linux kernel to
resolve the "use after free" flaw in the Xircom 16-bit PCMCIA
Ethernet driver.
- **Data Integrity Verification**: Implement strong encryption and
integrity checks in the Ethernet communication to detect tampering.
- **Enhanced Security Monitoring**: Monitor and log any anomalies in
data transmission and device communications.
- **AI Model Robustness**: Strengthen the ML model to detect and
handle potential adversarial inputs.
```

Figure 5: Generated causal scenario using ChatGPT 4o for the Ethernet communication in ABMD software

Name	Focus	Application Domain	Automation Level (Causal Scenarios)
A-STPA XSTAMPP	Safety	General Purpose	◐
SafetyHAT	Safety	Transportation	◐
WebSTAMP	Safety/Security	Healthcare, Transportation, Chemical Industry	◐
SOT	Safety/Security	Aircraft Systems	◐
SAM Our Tool	Security	Healthcare	●

Table 3: Summary of State-of-the-Art STPA/STPA-Sec tools (● : Fully-automated, ◐ : Semi-automated, ○ : Manual)

attacks on ML-enabled systems, whether they apply to the medical domain, and whether they automate the process of causal scenario generation.

To this end, we compare SAM with four state-of-the-art STPA/STPA-Sec tools: **(1)** A-STPA [1] and its enhanced version, XSTAMPP [2] – These tools help a user link unsafe control actions to identified safety hazards and provide graphical aids for control structure creation, but require manual identification of causal scenarios.; **(2)** SafetyHAT [4] – Customized for the transportation sector, this tool offers a graphical interface, data management capabilities, and transportation-specific guidewords for identifying unsafe control actions and causal scenarios, but still requires manual identification of comprehensive causal scenarios; **(3)** WebSTAMP [30] – It is a web application designed for STPA and STPA-Sec, that aims to provide a structured, automated, and comprehensive analysis. The tool offers a list of comprehensive control actions and guiding questions that help the user identify hazardous control actions and their causal scenarios. It has been demonstrated on a Glucose Monitoring and Insulin Pumping System, transportation applications[33], and chemical reactors[44]; and, **(4)** SOT [23] – This tool helps systems engineers conduct safety and security analyses by using stored knowledge from previous analyses to identify causal scenarios.

Table 3 shows a summary of these tools. A-STPA, XSTAMPP, and SafetyHAT focus only on safety concerns caused by device failures, not malicious data injection attacks. WebSTAMP and SOT address security concerns but rely heavily on users' knowledge of security vulnerabilities and manual effort. In contrast, SAM helps in gathering information on the latest security vulnerabilities from state-of-the-art databases, uses LLMs, and incorporates adversarial attack and medical domain knowledge, providing a comprehensive list of potential security risks for the system under assessment.

Recent papers such as the survey by Qi et al. [25] explore the use of STPA in learning-enabled systems, and introduce DeepSTPA for analyzing ML lifecycle failures, which is beyond our scope. Other recent papers [26, 20, 21] explore the usability of LLMs in STPA, highlighting the need for human intervention in generating prompts and validating LLM responses. However, they do not focus on medical device security, which is our focus.

6 Conclusion

This paper introduces SAM, a technique designed to assist in systematically performing STPA-Sec on ML-enabled medical devices. By identifying vulnerabilities in peripheral devices, SAM leverages the capabilities of LLMs to automate the identification of potential false data injection paths. We evaluated SAM using two case studies: a blood glucose management system and bone mineral density measurement software. The results demonstrated SAM's effectiveness in automatically identifying a comprehensive list of potential attack paths and their subsequent health impact on patients. With SAM, both device manufacturers and security analysts can identify security risks in ML-enabled medical devices at early stages.

In the future, we will explore simultaneous data injections at multiple vulnerable points, additional types of ML attacks, and develop mechanisms to summarize and automatically verify the outcomes generated by LLMs.

Acknowledgment

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the National Research Council of Canada's (NRC) Digital Health and Geospatial Analytics Program, and the U.S. National Science Foundation (CNS-2146295). We thank the ACM HealthSec'24 reviewers for their insightful comments.

References

[1] Asim Abdulkhaleq and Stefan Wagner. 2014. *Open tool support for system-theoretic process analysis.* Universitätsbibliothek der Universität Stuttgart.

[2] Asim Abdulkhaleq and Stefan Wagner. 2015. XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering.

[3] Alejandro Barredo Arrieta et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.

[4] Christopher Becker and Qi Van Eikema Hommes. 2014. Transportation systems safety hazard analysis tool (SafetyHAT) user guide (version 1.0). Tech. rep. USA.

[5] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*. PMLR, 2206–2216.

[6] CVE. 2024. CVE Search: Ethernet. (2024). Retrieved July 3, 2024 from https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Ethernet.

[7] CVE. 2024. CVE Search: Wi-fi. (2024). Retrieved July 3, 2024 from https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=wifi.

[8] CVE. 2024. CVE-2023-1670. (2024). Retrieved July 3, 2024 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-1670.

[9] CVE. 2024. CVE-2023-3583. (2024). Retrieved July 3, 2024 from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-35836.

[10] d-Nav. 2024. d-Nav by Hygieia. (2024). Retrieved July 3, 2024 from https://d-nav.com/.

[11] Mohammed Elnawawy, Mohammadreza Hallajiyan, Gargi Mitra, Shahrear Iqbal, and Karthik Pattabiraman. 2024. Systematically Assessing the Security Risks of AI/ML-enabled Connected Healthcare Systems. In *Proceedings of IEEE/ACM CHASE '24*, 97–108.

[12] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the ACM SIGSAC CCS '15*, 1322–1333.

[13] HeartLung. 2024. AutoBMD. (2024). Retrieved July 3, 2024 from https://www.heartlung.ai/autobmd-physicians.

[14] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems*. Vol. 36. Curran Associates, Inc., 51991–52008.

[15] Melissa Chase, Steven Christey Coley, Ronnie Daldos, and Margie Zuk. 2023. Next steps toward managing legacy medical device cybersecurity risks, The MITRE Corporation. (2023). Retrieved July 3, 2024 from https://www.mitre.org/news-insights/publication/next-steps-toward-managing-legacy-medical-device-cybersecurity-risks.

[16] Yisroel Mirsky, Tom Mahler, Ilan Shelef, and Yuval Elovici. 2019. CT-GAN: malicious tampering of 3d medical imagery using deep learning. In *USENIX Security '19*. USENIX Association, 461–478.

[17] Mitre. 2024. Common Vulnerabilities and Exposures (CVE) Database. (2024). Retrieved July 3, 2024 from https://cve.mitre.org/.

[18] Akm Iqtidar Newaz, Nur Imtiazul Haque, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. 2020. Adversarial Attacks to Machine Learning-Based Smart Healthcare Systems. In *IEEE GLOBECOM '20*, 1–6.

[19] NIST. 2024. National Vulnerability Database. (2024). Retrieved July 3, 2024 from https://nvd.nist.gov/.

[20] Ali Nouri, Beatriz Cabrero-Daniel, Fredrik Torner, Hakan Sivencrona, and Christian Berger. 2024. Welcome Your New AI Teammate: On Safety Analysis by Leashing Large Language Models. In *Proceedings of the IEEE/ACM CAIN '24*, 172–177.

[21] Ali Nouri, Beatriz Cabrero-Daniel, Fredrik Törner, Hakan Sivencrona, and Christian Berger. 2024. Engineering Safety Requirements for Autonomous Driving with Large Language Models. eprint: arXiv:2403.16289.

[22] Ollama. 2024. Ollama. (2024). Retrieved July 3, 2024 from https://ollama.com/.

[23] Daniel Patrick Pereira, Celso Hirata, and Simin Nadjm-Tehrani. 2019. A STAMP-based ontology approach to support safety and security analyses. *Journal of Information Security and Applications*, 47, 302–319.

[24] Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala Al-Fuqaha. 2021. Secure and Robust Machine Learning for Healthcare: A Survey. *IEEE Reviews in Biomedical Engineering*, 14, 156–180.

[25] Yi Qi, Yi Dong, Siddartha Khastgir, Paul Jennings, Xingyu Zhao, and Xiaowei Huang. 2023. STPA for Learning-Enabled Systems: A Survey and A New Practice. In *Proceedings of IEEE ITSC '23*, 1381–1388.

[26] Yi Qi, Xingyu Zhao, Siddartha Khastgir, and Xiaowei Huang. 2023. Safety Analysis in the Era of Large Language Models: A Case Study of STPA Using ChatGPT. eprint: arXiv:2304.01246.

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC*^{2} *Workshop*.

[28] Shubhra Kanti Karmaker Santu and Dongji Feng. 2023. Teler: A general taxonomy of llm prompts for benchmarking complex tasks. eprint: arXiv:2305.11430.

[29] Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, 623, 7987, 493–498.

[30] Fellipe GR Souza, Daniel P Pereira, Rodrigo M Pagliares, Simin Nadjm-Tehrani, and Celso M Hirata. 2019. WebSTAMP: A web application for STPA & STPA-Sec. In *MATEC Web of Conferences*. Vol. 273. EDP Sciences, 02010.

[31] Mehtab Tariq, Yawar Hayat, Adil Hussain, Aftab Tariq, and Saad Rasool. 2024. Principles and Perspectives in Medical Diagnostic Systems Employing Artificial Intelligence (AI) Algorithms. *International Research Journal of Economics and Management Studies*, 3, 1, 376–398.

[32] The Medical Futurist. 2024. FDA-approved A.I.-based algorithms. (2024). Retrieved July 3, 2024 from https://medicalfuturist.com/fda-approved-ai-based-algorithms/.

[33] John Thomas and Nancy G Leveson. 2011. Performing hazard analysis on complex, software-and human-intensive systems. In *Proc. of the 29th ISSC Conference about System Safety*.

[34] U.S. FDA. 2024. ABMD Software. (2024). Retrieved July 3, 2024 from https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K213760.

[35] U.S. FDA. 2023. Artificial Intelligence and Machine Learning (AI/ML)-Enabled Medical Devices. (2023). Retrieved July 3, 2024 from https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-aiml-enabled-medical-devices.

[36] U.S. FDA. 2024. CardioLogs ECG Analysis Platform. (2024). Retrieved July 3, 2024 from https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K170568.

[37] U.S. FDA. 2024. d-Nav System. (2024). Retrieved July 3, 2024 from https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K181916.

[38] U.S. FDA. 2024. NuVasive Pulse System. (2024). Retrieved July 3, 2024 from https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K180038.

[39] U.S. FDA. 2024. One Drop Blood Glucose Monitoring System. (2024). Retrieved July 3, 2024 from https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/pmn.cfm?ID=K161834.

[40] U.S. Food and Drug Administration (FDA). 2019. Premarket Approval (PMA). (2019). Retrieved July 3, 2024 from https://www.fda.gov/medical-devices/premarket-submissions-selecting-and-preparing-correct-submission/premarket-approval-pma.

[41] Zhibo Wang, Jingjing Ma, Xue Wang, Jiahui Hu, Zhan Qin, and Kui Ren. 2022. Threats to Training: A Survey of Poisoning Attacks and Defenses on Machine Learning Systems. *ACM Comput. Surv.*, 55, 7.

[42] Tahreem Yaqoob, Haider Abbas, and Mohammed Atiquzzaman. 2019. Security Vulnerabilities, Attacks, Countermeasures, and Regulations of Networked Medical Devices—A Review. *IEEE Communications Surveys & Tutorials*, 21, 4, 3723–3768.

[43] William Young and Nancy Leveson. 2013. Systems thinking for safety and security. In *Proceedings of ACM ACSAC '13*. Association for Computing Machinery, 1–8.

[44] William Young and Reed Porada. 2017. System-theoretic process analysis for security (STPA-SEC): Cyber security and STPA. In *STAMP Conference*. MIT Press, 27–30.