electronics

MDPI

Article

An IoT Featureless Vulnerability Detection and Mitigation Platform

Sarah Bin Hulayyil [1,2] and Shancang Li [1,*]

[1] School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, UK; binhulayyilsh1@cardiff.ac.uk
[2] College of Applied Studies and Community Service, King Saud University, Riyadh 11451, Saudi Arabia
* Correspondence: shancang.li@uieee.org

Abstract: With the increase in ownership of Internet of Things (IoT) devices, there is a bigger demand for stronger implementation of security mechanisms and addressing zero-day vulnerabilities. This work is the first to provide a platform that combines featureless approaches with artificial intelligence (AI) algorithms, which are deep learning and large language models, to uncover IoT security vulnerabilities based on network traffic data directly without manual feature selection. The platform correctly identifies vulnerable and secure IoT devices just by raw network traffic! Experimental results show that the proposed study detects vulnerability with great accuracy by using pre-trained deep learning and LLM models, which facilitates direct extraction of vulnerability features from the dataset and therefore helps speed up the identification process. In addition, the design of the platform ensures that the models are accessible and can be easily applied by users with a user-friendly interface. Furthermore, the models with small sizes, 277.5 MB and 334 MB for the deep learning model and the LLM model, respectively, illustrated the potential use of the detection tool in practical settings. The ability to defend large-scale, diversified IoT ecosystems efficiently and in a scalable way by installing thousands of software manifestations quickly while exposing new applications to growing cyber threats is made possible by this significant advancement in the field of IoT security.

Keywords: IoT security; LLM; deep learning; vulnerability detection

check for updates

Academic Editor: Aryya Gangopadhyay

Received: 9 February 2025
Revised: 24 March 2025
Accepted: 25 March 2025
Published: 4 April 2025

Citation: Hulayyil, S.B.; Li, S. An IoT Featureless Vulnerability Detection and Mitigation Platform. *Electronics* **2025**, *14*, 1459. https://doi.org/10.3390/electronics14071459

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The IoT has seen explosive growth in recent years [1]. It is now prevalent in many sectors such as smart homes, healthcare, manufacturing, critical infrastructure, etc. IoT devices offer high connectivity, automation, and data-driven functionality [2]. However, the widespread deployment of IoT devices introduces several security risks. Due to their interconnected nature and limited computational capabilities, IoT devices are more susceptible to cyber threats, including zero-day attacks—unknown vulnerabilities exploited by attackers before the vendor becomes aware or before a patch is released.

As IoT devices increasingly handle sensitive data and control critical systems, the need for robust security measures becomes essential. Current vulnerability detection methods for IoT systems rely on manually designed feature extraction, which has proven effective to some extent. Nonetheless, these methods possess inherent limitations in their ability to adapt to new or unknown threats. Feature engineering can be costly and time-consuming, and it may not effectively generalize to the constantly evolving threat landscape [3].

Recent advances in artificial intelligence (AI) and Machine Learning (ML) have created new opportunities to enhance IoT security [4]. AI-driven approaches, particularly deep

learning models, enable automatic learning of patterns or deviations in large-scale data without manual feature construction. This capability is valuable for uncovering hidden vulnerabilities often ignored by traditional detection methods.

Existing approaches to detecting IoT vulnerabilities often depend on manual feature engineering, which is time-consuming and highly susceptible to missing minor patterns in network data. Furthermore, many traditional solutions suffer from limited real-time detection capabilities and typically lack integrated mitigation strategies, necessitating separate procedures to address discovered vulnerabilities. These shortcomings create a clear need for a novel approach that can automatically extract features, operate effectively in real time, and provide end-to-end vulnerability identification and mitigation, a goal that our proposed strategy aims to achieve.

Given the importance of IoT devices in modern infrastructures and the increasing complexity of cyber attacks, there is a need for scalable, effective, and efficient solutions that can operate in real-world environments. Therefore, these solutions should be user-friendly and lightweight, as both devices and networks often have limited computational resources.

In our approach, we analyze network traffic directly, eliminating the need for manual data setup. This method simplifies the process and enhances the system's ability to respond quickly in practical settings. This combination of direct data processing, lightweight design, and built-in remediation distinguishes our approach from existing methods. We expect our paper to provide a clearer picture of the emerging risks in mobile networks. Thus, the study employs featureless detection models with AI processing, addressing scalability concerns while enhancing detection accuracy and flexibility. This results in a scalable, real-time detection and resource-efficient solution that operates effectively in IoT environments. This study thoroughly addresses the technical and practical aspects of IoT security, offering effective protection for increasingly connected systems against complex cyber threats. The main contributions are summarized as follows:

- This paper proposes a novel method for zero-day IoT vulnerability detection, using a deep learning model enhanced by pre-trained LLMs to automatically extract vulnerability features.
- A resource-constrained platform was developed based on the detection model for real-world scenarios, which is able to run a 277.5 MB deep learning model and a 334 MB LLM model, allowing users to quickly identify vulnerabilities in network traffic.
- We developed an end-to-end framework that goes beyond vulnerability detection, employing featureless CNN and CyBERT models, to provide automated mitigation strategies. This integrated approach enables real-time discovery and targeted mitigation of critical vulnerabilities, such as Ripple20 CVEs, significantly enhancing IoT network security.

This paper is organized as follows: A review of related work is presented in Section 2. The proposed methods, including our approach, architecture, and functionality, are detailed in Section 3. The validation, which encompasses the setup and design of the experiment along with the results of the proposed platform, is discussed in Section 4. Finally, Section 5 concludes with a discussion of future work.

2. Related Works

In AI, a featureless system or model operates without an explicit representation of features to transmit information through the system. This concept traces back to early work by Rodney Brooks, who advocated for a theory of intelligence that is embodied. This means that intelligence must interface directly with the world through perception and action, rather than relying on intermediary representations [5]. This approach enables the training of autonomous robots in the real world without specifying features. Featureless paradigms

offer several advantages; however, they can struggle in complex scenarios where nuanced understanding and representation could enhance performance. Brooks' stance highlights the ongoing discussion in AI regarding how to strike a balance between feature-based and featureless approaches.

2.1. Featureless in the IoT Security

Few previous works have utilized featureless approaches to enhance IoT security. These featureless methods are increasingly convenient for executing ML models in real time, particularly with regard to IoT security. These methods are increasingly convenient for executing ML models in real time, particularly in the context of IoT security. While previous studies have explored featureless approaches, most research has focused on specific sectors, such as healthcare and general security. A recent investigation introduced the FEL-ML (Feature Engineering-Less Machine Learning) model for malware detection in IoT architectures. One of the main limitations of existing methods is their reliance on extensive feature engineering, which entails manually crafting relevant features from raw packet data to optimize detection. In contrast, the FEL-ML model streamlines computations compared with traditional methods, which struggle with large datasets or require significant feature engineering.

In a recent study, a novel approach to monitoring and recognizing objects in dynamic environments was investigated. This method detects and monitors objects without relying on mathematical properties, contrasting with previous algorithms that often depend on such properties. The study demonstrated the robustness and dependability of this approach in dynamic contexts and confirmed its efficacy through real-world experiments and simulations. The results revealed that this featureless technique offers an effective method for item detection and tracking, particularly in challenging and dynamic applications [6].

2.2. Convolutional Neural Network (CNN) Models in IoT Security

CNN models have been widely used in IoT security, achieving excellent performance in recognizing and categorizing cyber attacks. Various studies demonstrate that CNNs enhance intrusion detection systems and anomaly detection in IoT environments. One study compared CNNs with traditional ML models and found that a CNN-BiLSTM model achieved accuracies of 99.89% and 98.95% on benchmark datasets, outperforming conventional methods [7]. Another study developed a CNN-LSTM model to classify IoT traffic, achieving 98.42% accuracy using a large dataset, highlighting its robustness against cyber threats [8].

Other studies have utilized CNN models in IoT security through anomaly detection and traffic classification. For example, CNNs combined with Variational Autoencoders (VAEs) are effective in detecting anomalies in IoT environments. This combination has achieved a classification accuracy of up to 95% [9]. CNNs have also successfully identified cybersecurity-related threats, achieving 99.10% for multiclass classifications and 99.40% for binary classification [10]. While CNNs deliver satisfying results in securing IoT by accurately detecting threats, hurdles remain. The primary issue is the computational complexity of running CNN models within the time constraints of real-time IoT systems. It is essential to note that models can quickly become outdated as cyber threats evolve, highlighting the need for continuous model updates. This motivates future research to develop low-complexity CNNs that are also resilient against new security threats [11].

2.3. LLM Models for IoT Security

The use of LLMs in security issues related to IoT has recently gained attention, with research efforts aimed at increasing cybersecurity levels and enhancing user comprehension of privacy policies. Although LLMs could have many applications in this context, some

studies indicate the need for new techniques to improve their effectiveness in other fields. LLMs have potential uses in IoT security across various domains, for example, Cybersecurity Helper: open-source LLMs that may assist in simulating vulnerable IoT systems and provide cybersecurity advice without disclosing personal data to cloud services or compromising user privacy [12]. Another application is Privacy Policy Simplification. Quantized LLMs are proposed to address the challenges of analyzing and simplifying complex IoT privacy policies in large-scale networks. This approach makes information more understandable to users while enhancing transparency and user-friendliness [13]. The final application is text-based generative IoT systems. A modified prompt approach has been suggested to improve the operational efficiency of open-source LLMs in local networks, making them more secure than commercial alternatives in IoT environments [14]. However, the use of LLMs in IoT security raises several concerns. First is corruption: quantized LLMs may be vulnerable to backdoor attacks. In this type of attack, adversarial behaviors can be injected into the model, enabling attackers to perform malicious actions while appearing to operate normally [15]. Second, protecting intellectual property is essential to prevent the piracy of LLMs deployed on edge; solutions such as SLIP have been proposed. These solutions maintain the privacy of the models to prevent theft and introduce a specific amount of overhead, allowing businesses to keep their white-box models secure [16]. Thus, even though LLMs show great promise in strengthening IoT security, their deployment requires careful arrangements to mitigate potential risks and vulnerabilities.

2.4. LLM Models in IoT Security Mitigation

Recently, LLMs have shown significant potential in detecting and preventing cyber attacks, particularly in the context of code generation and security audits. Their ability to analyze and generate code can help identify vulnerabilities, although challenges regarding accuracy and false positives remain. Several studies have used LLM to improve IoT security, for example, [17]; this work introduces LLMSecEval, a dataset of 150 natural language prompts aimed at evaluating the security behavior of LLMs when generating code. Each of the prompts relates to code that has vulnerabilities associated with problems specified in the Top 25 Most Common Software Security Errors (CWEs), which is a standard for the software community. To perform such a comparison, the authors made sure to include a dataset of secure code with all the prompts to have a proper standard for comparison. The application of LLMSecEval illustrates how it is useful in a more clinical manner to judge the security quality of code produced from NL instructions. Moreover, it also demonstrates the capabilities and shortcomings of such models in generating secure code. However, the context of the dataset is limited to the Top 25 CWEs. Although this is certainly thorough, it may ignore more recent or newer security issues that may be applicable to existing software systems. Furthermore, the paper does not use run-time or dynamic security evaluation of the code snippets, only static evaluation, and therefore misses the opportunity to identify additional vulnerabilities that static techniques would not identify.

A recent study [18] investigates how LLMs, such as GPTs, can be incorporated into system-on-chip (SoC) security verification to address issues of flexibility, scalability, and comprehensiveness. LLMs show promise by using language comprehension and program synthesis to enable effective and sophisticated security verification. To illustrate the advantages and challenges of this approach, the paper provides a comprehensive review of existing research, along with real-world case studies and experiments. However, there are several limitations regarding the use of LLMs in SoC security verification. Awareness of new risks identified after training is constrained by the knowledge cut-off point. Dependence on static analysis limits the detection of vulnerabilities that may emerge during run-time. Since LLMs are trained primarily on software and lack contextual knowledge,

they demonstrate insufficient coverage of hardware-specific vulnerabilities. Moreover, generalization issues hinder their ability to adapt to hardware-centric security requirements. Lastly, LLMs struggle to comprehend complex hardware behaviors and relationships, making it challenging to identify hidden security vulnerabilities in intricate systems.

2.5. Current Detection Platforms in IoT Security

Several previous works have designed platforms to enhance IoT security. In one paper [19], researchers highlight the necessary security and privacy issues in IoT, noting well-known vulnerabilities since most IoT devices belong to wireless sensor networks (WSNs). The paper presents a passive network traffic-based security intrusion detection model to monitor threats, including Denial of Service (DoS) attacks, brute-force attempts, and device tampering. This model uses sniffing tools to identify irregular patterns, unauthorized access, and unusual device activity or behavior. It incorporates mechanisms such as automated responses (alerts or device isolation), which, combined with an iterative feedback loop for continual improvement in threat detection, make it responsive to changes in attack types and vectors.

However, this study points out some limitations. The model may be harder to deploy due to real-time network monitoring, and approximations in congestion estimation using sniffing tools introduce privacy concerns. While these tools are ideal for security purposes, they can inadvertently capture sensitive information, leading to ethically concerning data privacy issues. Additionally, the complexities involved in deploying a successful network monitoring setup make real-world implementation challenging. Despite these challenges, the model presented illustrates an attempt to address IoT security from a proactive standpoint, balancing the need for robust intrusion detection with privacy considerations.

Secondly, in the study in [20], the researchers proposed an intrusion detection system (IDS) using Raspberry Pi for IoT devices to address this gap. The study aims to understand how effectively an IDS can identify various network attacks, enhance its configuration for improved performance, and evaluate its vulnerabilities. A simulated IoT environment utilizing the Raspberry Pi IDS, a temperature sensor, and background traffic generators is employed to conduct experiments and analyze system performance through metrics derived from log files. Finally, the expected outcomes of this research include demonstrating the concept of securing IoT devices with a Raspberry Pi-based IDS and providing recommendations on tuning the Snort IDS for optimal performance on a Raspberry Pi device.

However, securing IoT networks requires more comprehensive solutions than the specific lightweight Raspberry Pi-based intrusion detection system (IDS), particularly in larger or more diverse IoT environments. Additionally, IDS evaluation is limited to specific network attacks, which may hinder its effectiveness against novel or more sophisticated threats. It also depends on proper configuration and necessitates further investigation into the optimal settings. Finally, the results cannot be fully generalized to all IoT scenarios, as this study was conducted using a particular setup, indicating that special adaptations are needed for various IoT applications.

Harwalkar et al. addressed the major security issues facing IoT networks due to the large-scale data exchange between connected devices [21]. The authors proposed a neural network-based IDS that employs Long Short-Term Memory (LSTM) and Tuna Swarm Optimization to enhance intrusion detection accuracy in an IoT environment. The model utilized the NSL-KDD dataset, which was preprocessed using SMOTE for balancing and RFE for feature filtering. Feature selection was refined further through the Moth Flame Optimization (MFO) process. Next, they implemented the TSO-LSTM model for attack detection and used TSO optimization to identify attacks based on LSTM outputs, achieving a detection accuracy of 99.98%. This model outperformed several benchmark approaches,

including DBN, SMO-HPSO, CNN-LSTM, and BFO-RF, which achieved accuracies ranging from 98.8% to 99.96%. However, this study has limitations regarding the proposed model. Its reliance on the NSL-KDD dataset may restrict generalization, as it might not accurately represent realistic IoT environments.

Furthermore, the complex combination of LSTM and TSO in the model may be challenging to implement and tune in real-world applications. Another concern is that the system may not scale well and could perform poorly as the number of IoT devices increases. Furthermore, the study lacks sufficient information to identify which types of attacks intentionally targeting the model would be effective, limiting its performance against a broader range of threats. Finally, the absence of real-time testing overlooks a crucial aspect of the model's practical performance, significantly important in IoT dynamic environments that require real-time responsiveness.

Finally, the work in [22] proposed a real-time security monitoring (RSM) platform that utilizes deep learning models such as CNN, LSTM, and Deep Neural Networks (DNN) to predict and visualize attacks on IoT networks. The models are evaluated using the IoT23 dataset based on precision, recall rate, and F1-score values. The practical experiment employs a Raspberry Pi to collect log data. An edge router sends this data to the server, enabling real-time predictions, while Power BI serves as the dashboard for monitoring IoT performance. As a result, the RSM platform demonstrates improved performance for attack prediction.

Table 1 presents a comparative analysis of the latest related works and explains how this paper enhances them.

In conclusion, many works demonstrate that conventional methods, such as CNN or LSTM models, can effectively identify known security issues in network traffic. These studies have provided valuable insights into the effectiveness of data-driven detection in controlled settings. However, a major limitation of these approaches is their heavy reliance on manually defined steps to extract features from raw data. This process not only demands expert knowledge but also makes the systems less adaptable to new or evolving threats. Additionally, many methods face challenges when deployed in environments with limited resources, as they tend to be computationally intensive. Another gap in the existing literature is the lack of integration between detection and mitigation. While many methods focus solely on identifying vulnerabilities, few offer built-in solutions or recommendations to address these issues in real time.

Finally, several limitations affect the robustness and practicality of existing platforms. Deep learning-based models are central to many real-time security monitoring systems, but their performance can vary significantly depending on the dataset and type of attack used during testing. For example, the IoT23 dataset is only one instance and may not encompass all possible IoT attack methods, which can impact the overall robustness of the platform. Moreover, in massive IoT networks involving numerous devices, real-time processing can pose significant challenges. Additionally, issues with integration and standardization may arise when attempting to incorporate a real-time security monitoring platform into various existing IoT architectures and systems, leading to further complications.

As summarized in Table 2, our research addresses important gaps in the literature on IoT security and intrusion detection through an innovative, featureless AI-based methodology. Previous research has demonstrated that standard IDS is limited in IoT scenarios. For example, a low-resource Raspberry Pi-based intrusion detection system, optimized for IoT environments, has been introduced. However, its restricted scalability, dependence on specific network setups, and challenges in adapting to complex IoT ecosystems with diverse device types limit its effectiveness. Additionally, research has explored the use of

DL approaches for intrusion detection; however, these studies often rely on feature extraction techniques or complex optimization algorithms, which hinder real-time flexibility and increase implementation complexity. The limited capacity of these models to effectively address zero-day threats arises from their reliance on complex, manually created features to detect vulnerabilities.

Table 1. Comparative Analysis of Existing Works and Improvements in Our Study.

Reference	Approach	Dataset	Performance	Limitations	How Our Work Improves
[19]	Passive sniffing detects network anomalies and unauthorized access efficiently.	Controlled network traffic	Not specified	Sensitive data risk, real-time processing and integration complexities	Direct raw traffic processing with automated, real-time remediation solves privacy and deployment issues.
[20]	Raspberry Pi IDS detects simulated IoT network attacks.	Simulated IoT setup	Not specified	Scenario-limited, scalability issues, extensive configuration required	Lightweight, scalable, intuitive: ideal for diverse, resource-limited IoT deployments
[21]	LSTM neural network IDS with Tuna Swarm Optimization detects attacks.	NSL-KDD	Up to 99.98% detection accuracy	Generalization limitations, incomplete IoT attack dataset possible	Comprehensive framework: diverse IoT scenarios, real-time mitigation recommendations integrated
[22]	DL-based RSM platform predicts and visualizes attacks in real-time.	IoT23	0.86 for LSTM, 0.95 for CNN, and 0.99 for DNN	Restricted attack coverage, real-time scaling issues, diverse IoT integration challenges	Accurate, end-to-end vulnerability detection with tailored, real-time remediation.

In order to overcome these challenges, we have developed the first platform that combines featureless methods with DL and LLM models. These models examine unprocessed network traffic data to identify zero-day IoT vulnerabilities. In contrast to traditional methods that rely on manual feature extraction, our platform directly extracts and analyses vulnerability factors from the dataset using pre-trained AI models. This approach increases scalability and simplifies the detection process, allowing our lightweight models to perform efficiently even in IoT environments with limited resources. Furthermore, we integrate a user-friendly interface that addresses the accessibility gap noted in previous research, enabling non-technical individuals to utilize complex IoT security measures. Our study directly addresses the problems identified in earlier studies by offering a scalable, effective, and flexible solution for IoT security.

Table 2. Overview of the Proposed Method and Related Research.

Aspect	Previous Works	Our Approach
Methodology	Relied on manually defined steps to process data and detect threats	Processes raw network traffic directly without manual intervention, streamlining the detection process
Performance	Often faced challenges with slower processing times and lower adaptability to new threats	Offers faster detection with high accuracy and is able to adjust to evolving security challenges quickly
Real-World Application	Typically not designed for use in settings with limited resources and may lack user-friendly tools	Optimized for environments with limited computing power and includes an intuitive interface suitable for both experts and non-specialists
Integrated Mitigation	Often requires manual steps and separate tools for mitigation	Provides automated mitigation strategies alongside detection, offering an end-to-end solution within a single platform

3. Methodology

The featureless approach identifies features from raw input without requiring professionals to manually extract them, unlike classic ML techniques that depend on feature discovery as input for the model. Featureless approaches promise automatic feature learning from raw input. They utilize multiple layers of DL networks to build complex representations from the input. Compared with current DL-based models, featureless DL approaches can achieve superior performance, including enhanced accuracy [23,24].

The process of converting heterogeneous data types into an appropriate format is crucial but time-consuming in AI and ML applications, as models often require standardized numerical data. Traditional IoT security approaches often rely on manually extracting certain features from raw network traffic. This manual process requires significant expert involvement to identify and extract relevant data attributes. Moreover, it may overlook minor or emerging patterns outside of pre-established factors, leading to decreased accuracy and slower adaptation to new threats. In contrast, our approach analyzes raw network traffic directly, eliminating the need for manual feature extraction. This allows the system to automatically identify essential data patterns that might be ignored when using predefined features. Direct processing streamlines the detection pipeline and increases efficiency by removing the time-consuming and costly phase of manual data preparation. Additionally, by dispensing with manual feature selection, our approach can adapt more rapidly to evolving security threats, providing a more robust solution for real-world IoT environments. The feature engineering process may not be necessary for identifying cyberattacks and unusual activities in real time. This can help ensure that accurate and timely responses to cyber incidents are implemented.

Our work aims to create a tool that utilizes a featureless instance-based 1D-CNN model and an LLM model to protect networks from both internal and external attacks and to discover IoT vulnerabilities in an IoT environment. The primary reason for using a featureless approach is to handle data as they are, which saves time and enhances realism. This design choice also facilitates use by non-expert users, enabling them to secure their networks effectively. In contrast, functional engineering requires identifying features, removing them from unprocessed data, and then combining, expanding, and filtering the remaining data. Consequently, these feature engineering strategies tend to be extensive, complex, and non-emergent. The advantages of the featureless approach have become increasingly vital in the IoT ecosystem. A fundamental principle behind the featureless method is to convert data into bytes and input them into the model without altering the

original traffic data. As a result, the feature extraction procedure has been eliminated from the ML phases.

From a mathematical perspective, feature engineering in ML techniques involves transforming unstructured data into a set of features that computers can understand. This process is essential for effective model training. The development of featureless Machine Learning algorithms addresses the need for this transformation. Consider a conventional ML framework where a dataset $\mathcal{D}$ encompasses n instances, each delineated by m attributes. Mathematically, it can be expressed as a matrix $\mathbf{X}$ with dimensions $n \times m$. The goal of the procedure is to create a mapping function f that combines the target vector $\mathbf{Y}$ with $\mathbf{X}$

$$\mathbf{Y} = f(\mathbf{X}) \tag{1}$$

The featureless approach uses a series of byte vectors $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}\}$, each of which b_i represents raw data in byte form. The objective assigned to the model is to identify a function $g(\cdot)$ that transforms these byte vectors into the intended results $\mathbf{Y}$,

$$\mathbf{Y} = g(\mathcal{D}) \tag{2}$$

The function $g(\cdot)$ can analyze the raw data independently and extract patterns from the byte-encoded input. This approach is particularly useful when there is uncertainty about the most effective features or when the dataset has multiple dimensions. Using raw data as representation at the byte level, featureless ML allows the model to learn from the smallest unit of input. As a result, the need for manual feature extraction is eliminated, potentially leading to more accurate models that are less susceptible to biases in the relative values of numerous features detected by humans.

As previously known, this work presents a featureless IoT vulnerability detector. We designed and programmed a tool using Python in Visual Studio that allows users to monitor their network traffic in real-time to ensure that both the network and connected devices are secure. The key aspect of this platform is its design, which is based on two trained models. Both models utilize featureless techniques, making them more accessible to users. The trained models are the 1D-CNN feature model and the featureless CyBERT model. Figure 1 illustrates how the IoT vulnerability detector platform functions at each step. It begins by uploading the network traffic file, followed by selecting one of the featureless models to analyze the provided network traffic. Once the model completes the prediction, it generates results that include several outputs to enhance accuracy and comprehensibility: detection time, vulnerability percentage, a chart showing the number of vulnerable and normal packets, and detailed information for each packet indicating which exact packets are vulnerable. In the following sections, we discuss how each model operates for IoT vulnerability detection.

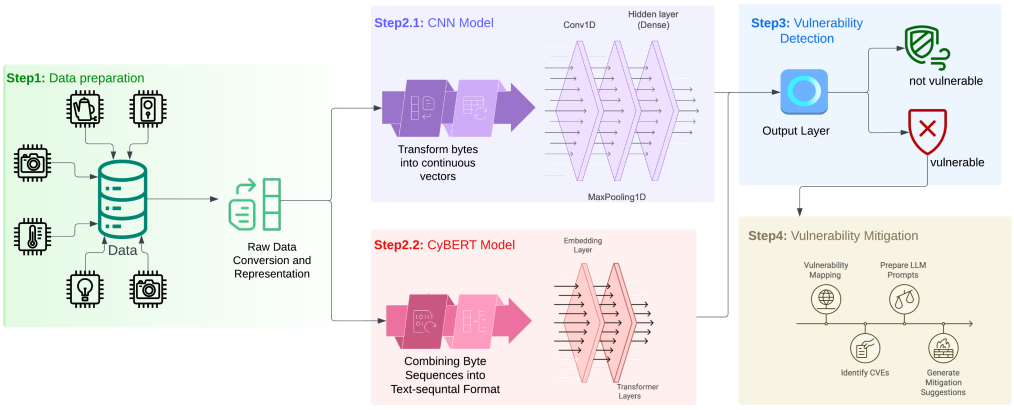

Figure 1. The featureless IoT vulnerability detection tool workflow.

3.1. Data Preparation

In Figure 1, **Step 1** shows how to prepare the data after collecting them from IoT devices. There is a single procedure in this process that employs Python scripts to transform network packets into bytes for processing by the model, in line with the featureless model approach. IoT devices generate various types of data, among which we use the network traffic data for our study. The featureless approach converts this raw data directly into byte sequences. A sequence of bytes represents every sample $\mathbf{x}_i$ of IoT data.

$$\mathbf{X}_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \tag{3}$$

where x_{ij} is the byte in the i-th sample at position j-th, and n is the total number of bytes in each sample. After that, if the CNN model is chosen, the data will progress to the input layer, which is used in the embedding layer to convert the bytes into continuous vectors. If the CyBERT model is chosen, these byte sequences are merged to form a single text-like sequence that the CyBERT model can process. The byte sequences are combined to create a continuous form that simulates text. This method ensures the model processes incoming data in a manner suitable for applications involving natural language processing. The combined sequence $\mathbf{S}_i$ for the i-th sample in a set of m samples is expressed as follows.

$$\mathbf{S}_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \tag{4}$$

The system processes raw data via two parallel pathways: a user-selectable 1D-CNN or CyBERT model. The following outlines the workflow for each processing branch.

3.2. Featureless CNN Model Architecture

To identify the significant risks of IoT vulnerability across the network, this model develops a straightforward, lightweight, featureless one-dimensional convolution neural network (1D-CNN) model that can be implemented in small devices, such as IoT devices and routers. In Figure 1, **Step 2.1** shows how the data are handled to be used in the CNN model, while the model details are illustrated in Figure 2. It can be summarized as a 1D-CNN with two convolutional layers, a fully connected layer, and a hidden layer. Since network packets entering the device can be viewed as cases dispersed over time, this scenario can be classified as a time series issue. One-dimensional CNNs have been applied in several previous studies to label time series data [25–27].

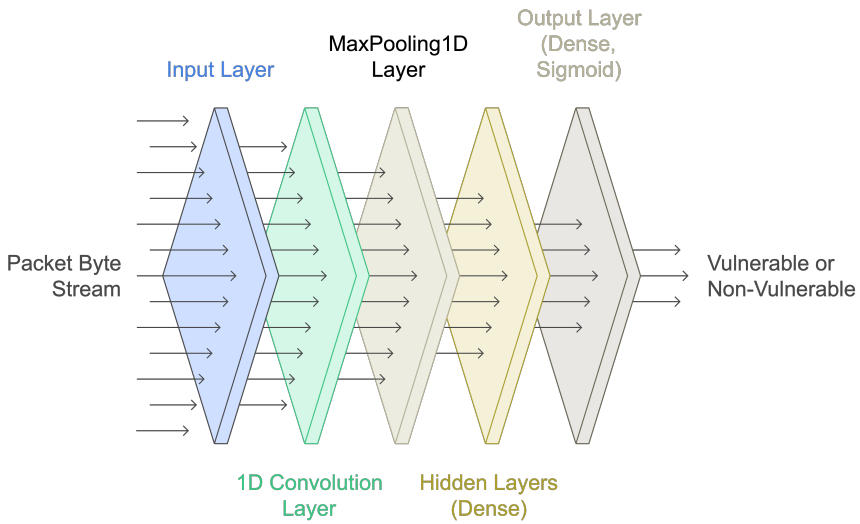

Figure 2. The featureless CNN model architecture.

3.2.1. Embedding of Byte Sequence

Bytes are transformed into continuous vectors in the embedding layer through the input layer, which represents the relationships and contexts between the vectors. These vectors are then fed into CNN layers, which utilize different types of layers to learn from higher-level feature instances. Each byte x_i is mapped to a continuous vector using an embedding function $e(\cdot)$. Thus, the embedded representation as

$$\mathbf{E} = [e(x_1), e(x_2), \ldots, e(x_n)] \tag{5}$$

3.2.2. Convolution Operation

The Conv1D layer follows the input layer. A one-dimensional convolutional filter of size k is applied to the embedded sequence. At a given position i, the convolution is computed as

$$z_i = \sigma\left(\sum_{j=1}^{k} w_j \cdot e(x_{i+j-1}) + b\right) \tag{6}$$

in which w_j are the filter weights, b is the bias, and $\sigma(\cdot)$ is an activation function.

3.2.3. Pooling and Classification

Then, the MaxPooling1D layer procedure receives the output after that to reduce the output size. Next, we implemented a single hidden layer that was built in order to teach the non-linear link to the model weights. A maxpooling operation is applied to the sequence $\{z_i\}$ to reduce its dimensionality as

$$p = \max\{z_1, z_2, \ldots, z_{n-k+1}\} \tag{7}$$

The pooled feature p is then passed through one fully connected layer to produce the final vulnerability score as

$$y = f(p) \tag{8}$$

in which $f(\cdot)$ denotes the mapping implemented by the fully connected layers.

Finally, the output layer consists of a Dense layer with a "sigmoid" activation function. This function is suitable for a prediction model, as its probability ranges only between 0 and 1 [28]. Since our model uses binary classification, we employed the Adam optimizer with a learning rate of 0.0001 and binary cross-entropy as the loss function while compiling the model to prevent overfitting. The model parameters for each layer are shown in Table 3.

Table 3. Featureless CNN model parameters.

Layer	Operation	Filter	Output
0	conv1D	6 × 200	5 × 200
1	Maxpooling1D	2 × 200	3 × 30
2	Dense	-	98,334
3	Dense	-	1

3.3. Featureless CyBERT Model Architecture

In Figure 1, **Step 2.2** shows how the data are handled to be used in the CyBERT model, while the model details in Figure 3 show how the featureless CyBERT model works in each step.

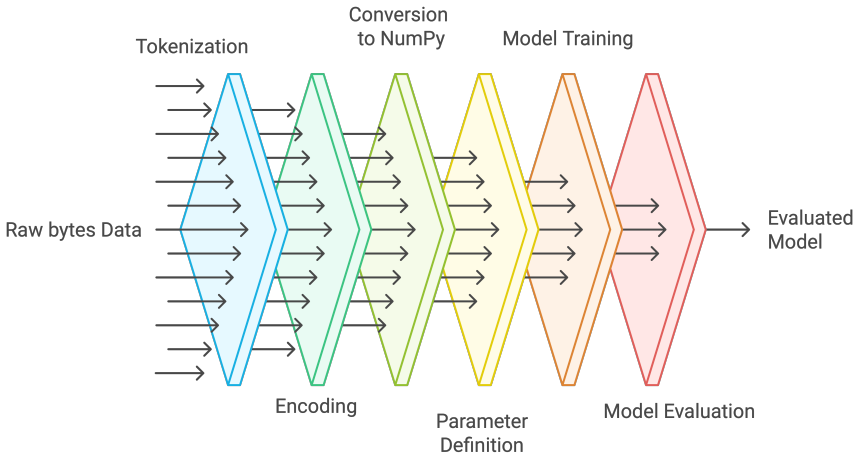

Figure 3. The featureless CyBERT model architecture.

3.3.1. The CyBERT Model Architecture

In Figure 3, the operation of the featureless CyBERT model at each step is illustrated. To address cybersecurity needs, the BERT architecture was modified to create the CyBERT model, which effectively analyzes raw byte sequences.

The process begins with the raw bytes of network traffic, and the model takes these raw bytes directly, capturing all information from the original data. Second, tokenization, which converts bytes into tokens, this tokenization makes the raw data compatible with subsequent text-like processing. Third, the tokens are then encoded into numerical vectors, preparing them for input to the model. This step translates the text-like representation into a form that can be handled by numerical computations. Fourth, the data are organized into NumPy arrays. This format is commonly used in many computational libraries, ensuring efficient data manipulation and integration with deep learning frameworks. Fifth, the internal parameters of the model, such as weights, biases, and learning rate, are set or initialized. This setup lays the foundation for training. Finally, during training, the model iteratively adjusts its parameters by examining labeled data. It refines its ability to identify potential vulnerabilities by minimizing the difference between its predictions and known labels. The performance of the model is then evaluated using separate data. This evaluation measures how well the model identifies vulnerabilities in previously unseen traffic.

The model architecture consists of several transformer layers designed to capture the relationships and patterns in the data. It includes the embedding layer and transformer layers.

The embedding layer: All bytes x_{ij} are assigned to a high-dimensional vector by an embedding function $e(x_{ij})$. The embedded sequence of the i-th sample can be identified by

$$\mathbf{E}_i = [e(x_{i1}), e(x_{i2}), \ldots, e(x_{in})] \tag{9}$$

Transformer layers: Several transformer layers that leverage feedforward neural networks and self-attention processes handle the encoded sequence. Let $\mathcal{T}$ be the transformation that these layers apply, and let $\mathbf{H}_i$ be the hidden representation of the sample for the i th sample that comes after the last transformer layer.

$$\mathbf{H}_i = \mathcal{T}(\mathbf{E}_i) \tag{10}$$

3.3.2. Processing Raw Traffic Data with CYBERT

The featureless CyBERT model processes network traffic by utilizing its language-understanding capabilities after converting the raw data into a text-like format. This section explains the step-by-step data conversion and fine-tuning process.

First, each network packet is converted into a sequence of bytes in Equation (3).

Next, the byte sequence X is transformed into a text-like sequence by joining the individual bytes. This process results in a string that captures key aspects of network traffic, e.g.,

$$S = \text{join}(X) \Longrightarrow \text{“Protocol: TCP; Port: 80; Packet Length: 1500 bytes; Flow Duration: 120 ms”} \quad (11)$$

The text sequence S is then used as input to the pre-trained CyBERT model. In our approach, the pre-trained CyBERT is fine-tuned on our labeled dataset to adapt it for IoT vulnerability detection. Fine-tuning adjusts the model parameters to ensure accurate interpretation of these text-like sequences and classification based on the presence of vulnerabilities. The expected output is a classification indicating whether the input exhibits signs of vulnerability. For example, the output might be “1”, where 1 indicates a detected vulnerability and 0 indicates no vulnerability.

3.4. Vulnerability Detection

To identify vulnerabilities, the output layer, known as the classification layer, receives the hidden representation $\mathbf{H}_i$, which estimates the probability that a sample is vulnerable. Unusual behaviors indicating vulnerabilities in IoT data can only be detected through vulnerability detection. The training procedure for both models aims to develop representations that distinguish between normal and vulnerable data. From the input data, the $g(\mathbf{S}_i)$ function generates an anomalous score. The objective of the vulnerability detection loss, which is the total of the losses $L_{\text{vulnerable}}$ for all training samples m, is to enhance the difference between the normal and vulnerable sample sets.

$$\mathcal{L}_{vul} = \sum_{i=1}^{m} \ell(g(\mathbf{S}_i), t_i) \quad (12)$$

where ℓ is used to identify vulnerabilities. To prevent the model from overfitting, the loss function ℓ is employed, with learning rates of 0.0001 for the CNN model and 1×10^{-5} for the CyBERT model. This function compares the anomaly score for the i-th sample, produced by the model $g(\mathbf{S}_i)$, with the ground truth label t_i.

3.5. Vulnerability Mitigation

Using LLM models to mitigate IoT vulnerabilities is complex due to the diversity of IoT data, as each threat requires different mitigation strategies. However, we designed these models to automate the identification and mitigation of network vulnerabilities by referencing Common Vulnerabilities and Exposures (CVEs), specifically from the Ripple20 vulnerability set, and utilizing LLM models to generate additional mitigation strategies. This approach is particularly valuable in a real-time network security context, where packets or network traffic must be evaluated for vulnerabilities and appropriate mitigations need to be applied swiftly. In our case, when the CyBERT model detects a vulnerability in the network traffic, the affected traffic is passed to the LLM model, specifically GPT-3.5, which identifies and analyzes the vulnerability to suggest the best solution. The mitigation step can be summarized as follows: the LLM receives the vulnerable packets, analyzes them to extract main features and patterns, and utilizes featureless models to identify essential features related to the vulnerability patterns, such as ports, protocol, and packet length, while also calculating the flow duration for each packet. The following section explains the steps for the mitigation suggestion and illustrates them mathematically.

Let us assume these notations (Table 4).

Table 4. Table of Notation.

$\mathcal{C}$	A set of known CVE entries relevant to Ripple20 vulnerabilities, e.g., $\{c_1, c_2, \ldots, c_k\}$.
$\mathcal{M}$	A set of possible mitigation strategies returned by the LLM.
$\mathcal{T}$	The space of textual prompts or user-provided messages/logs.
$\mathcal{V}$	A set of vulnerabilities detected in the IoT data (e.g., from a CyBERT model).
$\mathbf{y}_{\text{predicted}}$	$\mathbf{y}_{\text{predicted}} \in \{0,1\}^n$ is a binary vector of length n, where each entry corresponds to a packet classified as non-vulnerable (0) or vulnerable (1).
$\mathcal{P}$	$\mathcal{P} = \{p_1 \ldots p_n\}$ is a set of network packets or logs to be evaluated for vulnerabilities.

3.5.1. Implementation of LLM-Based Mitigation

Define vulnerability mapping with the main features, mapping the top vulnerabilities associated with IoT networks based on the Common Vulnerabilities and Exposures (CVEs) website; in our case, we focus on the Ripple20 vulnerabilities. The objective is to associate particular vulnerabilities (CVE entries) with their relevant attributes, enabling the following:

- The recognition of vulnerabilities through network characteristics such as protocol, port, and flow duration.
- The examination of possible attack vectors by utilizing the Common Vulnerability Scoring System (CVSS) score and comprehensive descriptions.
- The application of mitigation tactics specifically designed for each vulnerability.

Let ripple20_cve_mapping be a structure that associates each CVE ID $c \in \mathcal{C}$ with specific vulnerability details. We can denote this mapping as

$$M : \mathcal{C} \rightarrow \mathcal{V}_{\text{info}} \tag{13}$$

where $\mathcal{V}_{\text{info}}$ is the space of vulnerability details (e.g., CVSS score, description, and recommended patch). For each $c \in \mathcal{C}$,

$$M(c) = \{\text{description}, \text{CVSS\_score}, \text{attack\_vector}, \ldots\} \tag{14}$$

identifies matching vulnerability CVEs by comparing attributes of individual packets against the pre-defined Ripple20 vulnerability mapping. Then, prepare prompts for the LLM by scanning the provided text (prompt) for mentions of CVEs from the CVE mapping list. Return the corresponding vulnerability details if a match is found. This process is useful for identifying which CVE corresponds to specific information, such as protocols and ports.

Finally, generate structured, human-readable prompts summarizing packet details and linked vulnerabilities to query the LLM, associating the vulnerabilities with appropriate mitigations.

Generating a contextual prompt for LLM to produce mitigation strategies. Each vulnerability instance is distilled into a structured prompt that includes the relevant CVE details, protocol, port, flow duration, and preliminary mitigation steps. An example prompt is shown below Listings 1 and 2:

Listing 1. LLM Prompt Example for IoT Vulnerabilities.

```
prompt = f"Vulnerability: {cve_data['CVE']} - {cve_data['description']}\n" \
f"Protocol: {cve_data['protocol']} Port: {cve_data['port']} Flow Duration: {cve_data['flow_duration']}\n" \
f"Suggested Mitigation: {cve_data['mitigation']}\n\n" \
f"Now, Find an appropriate CVE for this IoT vulnerability if possible, rank the potential risk CVSS based on the CVE website and then Suggest mitigation."
]
```

This prompt is then provided to the LLM, along with a system message defining its role as a cybersecurity expert:

Listing 2. LLM message for the LLM.

```
messages = [
{"role": "system", "content": "You are a cybersecurity expert."},
{"role": "user", "content": f"Suggest mitigation for this IoT vulnerability: {prompt}"}
]
```

The system message ensures that the model aligns its response with security-focused best practices, while the user message presents the vulnerability details identified in our pipeline.

Mathematically, we define a function that scans a given text prompt $T \in \mathcal{T}$ for references to CVE IDs in $\mathcal{C}$. Formally,

$$\text{find\_matching\_cve} : \mathcal{T} \to \mathcal{C} \cup \{\varnothing\} \tag{15}$$

in which $\varnothing$ indicates that no match was found. Thus, for a prompt T, as

$$c = \text{find\_matching\_cve}(T) \tag{16}$$

If $c \neq \varnothing$, we retrieve additional vulnerability details from $M(c)$.

Generate mitigation suggestions for network vulnerabilities by passing descriptive prompts about flagged packets. It works by using packet-level vulnerability information encapsulated at the prompt to query an LLM. First, an advanced API provides additional strategies to mitigate detected vulnerabilities. If a prompt, such as a network log, includes a specific vulnerability reference, it is adjusted to request detailed mitigation suggestions and a summary of the associated information. This ensures a tailored and relevant response to the vulnerability in question. Second, to determine specific situations where vulnerabilities are present, a thorough examination of the anticipated vulnerability labels is carried out. The process described above enables a comprehensive approach to solving potential security threats, producing accurate mitigation methods for each case. Additionally, these recommendations should be included in the dataset for further reporting or analysis. Define

$$\text{get\_mitigation} : (\mathcal{T}, \mathcal{C}) \to \mathcal{M} \tag{17}$$

such that for a prompt T (e.g., a packet log or message) and a CVE ID c matched by find_matching_cve, the function constructs a specialized prompt for the LLM. This prompt requests remediation guidance for c based on the associated details $M(c)$, as

$$\mathcal{M}_c = \text{get\_mitigation}(T, c), \tag{18}$$

in which $\mathcal{M}_c$ is the mitigation strategy set suggested by the LLM. If no CVE is found, it may return a default or must relate to a mitigation set. Let $\mathbf{y}_{\text{predicted}}$ be the output

of a vulnerability detection model (e.g., CyBERT), indicating which packets are vulnerable. The function mitigate_vulnerabilities processes each packet p_i that is classified as vulnerable ($y_{\text{predicted},i} = 1$) and obtains mitigation suggestions via get_mitigation. As

$$\text{mitigate\_vulnerabilities} : (\mathcal{P}, \mathbf{y}_{\text{predicted}}) \rightarrow \mathcal{R} \tag{19}$$

in which $\mathcal{R}$ is the resulting set of records containing vulnerabilities, CVE mapping, and mitigation steps. The process is summarized as follows:

When submitted to the LLM, the model interprets the contextual input, which includes the CVE number, protocol, port, flow duration, and existing mitigation. It then returns a refined or additional mitigation plan.

(1). For each packet p_i:

$$\text{if } y_{\text{predicted},i} = 1, \text{ then gather relevant details from } p_i \text{ to form } T_i \in \mathcal{T}.$$

(2). Identify CVE IDs mentioned in T_i:

$$c = \text{find\_matching\_cve}(T_i).$$

(3). If $c \neq \varnothing$, generate mitigation suggestions:

$$\mathcal{M}_c = \text{get\_mitigation}(T_i, c).$$

(4). Aggregate results into a record ρ_i, associating $(p_i, c, \mathcal{M}_c)$.

$$\rho_i = \{\text{packet\_id} : p_i,\ \text{cve\_id} : c,\ \text{mitigation\_recommendations} : \mathcal{M}_c\}.$$

Finally, the set of records $\{\rho_i\}$ is returned as $\mathcal{R}$, providing a comprehensive mapping of detected vulnerabilities and suggested mitigation.

3.5.2. Rationale

- Context preservation ensures the model has complete context to produce accurate mitigation techniques by integrating protocol, port, and flow length with current vulnerability specifics (CVE, description, CVSS).
- In the expert role, "You are a cybersecurity expert", the system message directs the LLM to concentrate on threat analysis and security suggestions.
- With CVE alignment, requesting the model to "Find an appropriate CVE" checks the identified vulnerability against the model's knowledge base and may reveal additional threats.
- With adaptive responses, the LLM can comment on partial matches and suggest more comprehensive mitigations in the event of partial CVE mappings or new threats.

3.5.3. Verification of LLM-Based Mitigation Strategies

To enhance confidence in mitigation solutions generated by LLMs, we propose several verification approaches. Firstly, cross-validation with established databases, which are mitigation recommendations, is verified against well-known vulnerability databases and established fixing techniques. Secondly, rule-based post-processing. Before deploying the LLM results, we apply a set of predefined guidelines that ensure compatibility with industry best practices.

These verification procedures aim to reduce the likelihood of inaccurate or unactionable suggestions. They will be used to assess the mitigation framework in real-world circumstances.

4. Validation

This work aims to deploy the trained featureless models from the previous works. As the training results were outstanding, we created a platform that could help the user, expert or not, to ensure their IoT network. Both featureless models were trained on collected data from our lab and a set of IoT23 datasets, and both were used to detect and identify vulnerabilities related to IoT devices.

4.1. Datasets

The first dataset for our experiments consists of network traffic collected from a smart home network at Cardiff University Smart Home Lab. The network comprises 12 smart home devices, 6 of which are affected by Ripple20 vulnerabilities, while 6 are confirmed to be secure. The status of each device was determined by scanning it with the Nessus platform to identify the presence or absence of Ripple20 vulnerabilities.

The network packets were captured using Wireshark over a period of four hours. This duration was chosen because the Ripple20 vulnerability typically manifests during the initial connection phase when the device begins its authentication process. This method ensures that, even with a high volume of packets, critical vulnerability patterns are effectively captured. In total, the capture resulted in 253,914 packets saved in a Pcap file, which has a size of 324.2 MB.

The collected data were analyzed and labeled to distinguish traffic containing Ripple20 vulnerabilities from traffic that is free of these vulnerabilities. This labeling process enabled the model to learn and recognize the patterns associated with vulnerabilities during training.

The second dataset is a subset of the IoT23 dataset, specifically linked to our Ripple20 vulnerability case study and its implications regarding attacks. The IoT23 dataset comprises normal packets and DDoS attack packets, totaling 1,760,865 network packets with a size of 1.8 GB.

4.2. Training Configurations and Hyperparameters

Data were initially split into training and test sets using an 80/20 ratio with a fixed random seed *random_state* = 42 to ensure reproducibility. Within the training set, we used a validation split of 10% during model training to monitor and tune performance.

To ensure reproducibility and robust performance, we provide the key training configurations used for our models. Additionally, to address the class imbalance, we applied a *RandomOverSampler* to the training data only to ensure that the model learns effectively from minority classes without being biased towards the majority class. The test set, on the other hand, is left unaltered to accurately reflect the real-world distribution and to provide a fair evaluation of the model's performance.

Table 5 shows the training configurations and hyperparameters. The tuning was performed using grid search on a validation set. Additional parameters, such as dropout rates and weight decay, were adjusted based on preliminary experiments to prevent overfitting and improve generalization.

Table 5. Training Configurations and Hyperparameters.

Parameter	1D-CNN Model	CyBERT Model
Optimizer	Adam	Adam
Learning Rate	0.0001	1×10^{-5}
Batch Size	64	32
Number of Epochs	50	30
Loss Function	Binary cross-entropy	Binary cross-entropy

4.3. Experimental Setup

The featureless proposed model was evaluated by calculating various matrices and training and detecting time on different non-featureless LLM models such as BERT, SecureBERT, CySecBERT, and CyBERT. The results were then compared with those from our featureless model. Table 6 compares the performance, including training and validation time, for representative non-featureless techniques with our proposed featureless model. It is clear that our featureless CyBERT achieves excellent results within a shorter time for training and validation. This high level of efficiency is crucial in IoT deployments, where reducing power usage and providing immediate threat detection are top priorities. Although CySecBERT achieved comparable accuracy, it required nearly seven times as long to complete its training phase, illustrating the practical compromise between complexity and limited device resources. In addition, these outcomes align with previous research that underscores the importance of lightweight, near-real-time security measures in connected environments. So, we decided to use our featureless model within the detection and mitigation platform.

Table 6. Comparison of representative non-featureless techniques with our featureless model.

The Model	Accuracy	Precision	Recall	F_1-Score	Training and Testing Time (s)
BERT Model	0.9988	0.9988	0.9988	0.9988	11,724.72
SecureBERT Model	0.9979	0.9979	0.9979	0.9979	11,306.92
CySecBERT Model	0.9990	0.9990	0.9990	0.9990	11,697.347
CyBERT	0.9951	0.9951	0.9951	0.9951	5945.0
Featureless CyBERT	**0.999**	**0.999**	**0.999**	**0.999**	**1470.81**

For training the models, we conducted the experiment on an Ubuntu 22.04.1 LTS server, powered by an NVIDIA GeForce RTX 3080 with 32 GB of memory and a 12th-generation Intel i7-12700 CPU. Then, we used Visual Studio and Python to create the platform. Of the data collected, six IoT devices were linked to the Ripple20 vulnerability, while the remaining six were not affected. DDoS attack packets and regular packets made up the IoT23 dataset that was used.

4.4. Experimental Design and Rationales

As we evaluate the performance of these models, we calculate several metrics while considering the processing time. Table 7 shows the results of the metrics for both models. Accuracy serves as a training and validation measure for the data, as it is defined as the percentage of correctly categorized data among all classifications performed by the model.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

in which TP denotes a *true positive*, TN denotes a *true negative*, FP denotes a *false positive*, and FN denotes a *false negative*, respectively.

Table 7. Featureless CNN and CyBERT Models results.

Model	Accuracy	Precision	Recall	F_1-Score	RMSE	AUC-ROC	Time
Featureless CNN	0.9996	99994	0.99912	0.99953	0.02	0.99	837.57
Featureless CyBERT	0.999	0.999	0.999	0.999	0.0	1.0	1470.81

Precision is the ability of a classification model to identify and emphasize the most significant data points. Mathematically, it is defined as the ratio of true positives to the total number of true positives and false positives combined.

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

Recall is a mathematical term that describes a model's ability to find all relevant examples during data collection. It is expressed as the ratio of the total number of true positives to the sum of true positives and false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{22}$$

The F_1 score is calculated using the weighted average of accuracy and recall levels of the verified data.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{23}$$

A common method to quantify the differences between predicted and actual values is the Root Mean Square Error (RMSE). RMSE is defined as the square root of the mean of the squared differences between the actual and predicted values. This metric is especially useful for highlighting large errors, which are generally undesirable. It can be calculated using the following formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{24}$$

in which the following are used:

- n is the number of observations.
- y_i is the actual value for the i-th observation.
- $\hat{y}_i$ is the predicted value for the i-th observation.

4.5. Results

Table 7 presents two methods, the featureless CNN and the featureless CyBERT. Both methods achieve exceptionally high accuracy 99.96% and 99.9%, respectively, indicating that nearly all instances are correctly classified. In terms of precision, recall, and F1-score, both methods are similarly strong, suggesting that they not only correctly identify most positive cases but also avoid excessive false alarms. The RMSE is low in both approaches, and the AUC-ROC values reveal a high level of distinction between normal and vulnerable traffic. A key difference lies in processing time. The featureless CNN requires 837.57 s, whereas the featureless CyBERT needs 1470.81 s. Although both methods provide robust detection, the shorter time of the featureless CNN may be more practical for settings where devices have limited power or where fast responses are necessary. On the other hand, those willing to invest more time in processing may find the featureless CyBERT method attractive, especially since it achieves a perfect AUC-ROC score. Ultimately, the choice between these methods would depend on the specific needs and constraints of the intended application.

Since both models were trained on different datasets to distinguish between normal and vulnerable traffic, we are preparing to deploy these models for user access.

Figure 4 shows the platform interface that detects vulnerabilities through IoT network traffic. This platform takes several steps to present a user-friendly interface, allowing any user to deploy both models:

1. Model Preparation: Preparing the models for use after training is the first step. The models need to be stored in a format that supports easy integration into the tool after it has been trained. The most common format is h5 for the TensorFlow/Keras models that were used. By saving the model, it makes sure that it can be loaded and used in many scenarios, such as the Python-based use which is being developed.
2. Integrating the Model: The model must then be integrated into the Python tool that was being developed after it is prepared. While many more techniques might be employed, we choose to use API-based integration in our case. Using Python, the easiest way is to use packages like pickle and TensorFlow/Keras to import the stored model straight into the source code. As a result, the tool can obtain predictions from the API.
3. Building the User Interface: The next stage after integrating the model is to design a simple user interface so that participants can communicate with it. Users can provide data, see results, and obtain model predictions using a graphical interface designed with Python frameworks such as PyQt and Tkinter. This ensures that users can effectively use the possibilities of the model by making the tool easily accessible and user-friendly.
4. Testing and Optimization: Thorough testing and optimization are essential when the model is implemented and the interface is developed. This process makes sure that the model runs smoothly within the tool, that the predictions are correct, and that when users interact with the interface, it behaves as expected. The tool's dependability in practical situations is confirmed by testing under different scenarios. Performance optimization is also required to reduce memory usage, increase interface responsiveness, and improve prediction speed. This preserves the fact that the tool works well even when handling large datasets or producing predictions in real-time.

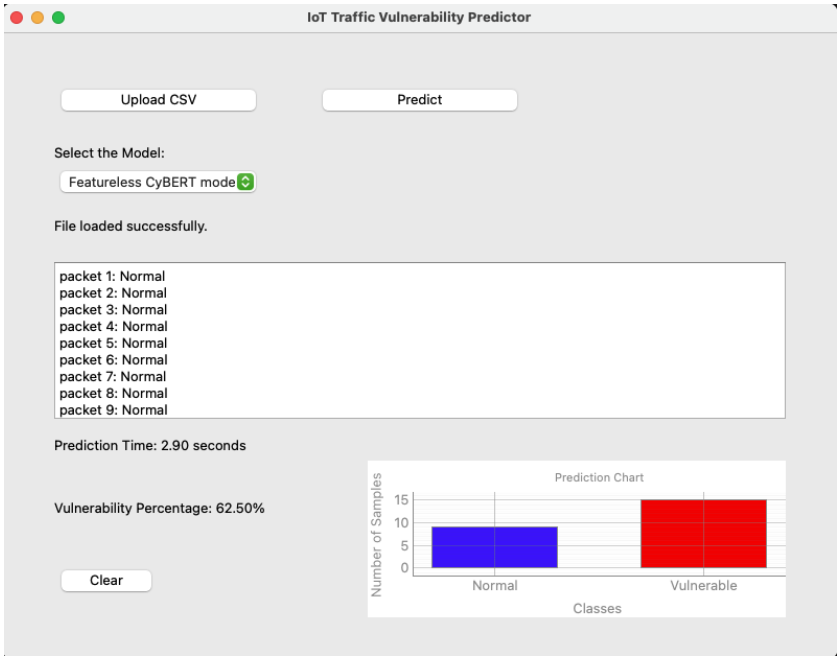

Figure 4. The featureless IoT vulnerability detector.

The figure illustrates that the detection tool is user-friendly for all users. When users upload a network traffic file, the tool detects vulnerable packets and presents various results, including detection time, vulnerability percentage of the packet file, total counts of vulnerable and normal packets, and additional details for each packet. Moreover, security experts can explore each packet further to gain deeper insights into the network traffic.

Then, when vulnerabilities are detected, the mitigation option will be activated. The vulnerable packets will be automatically sent to GPT-3.5 for mitigation suggestions based on the identified steps. The suggestions will be received in a few seconds, as shown in Figure 5. In conclusion, the tool is designed to automate the detection, identification, and mitigation of vulnerabilities in IoT networks by referencing known CVEs and utilizing AI to generate additional mitigation strategies. This approach is valuable in a real-time network security context, where packets or network traffic must be evaluated for vulnerabilities, and appropriate mitigation applied immediately.

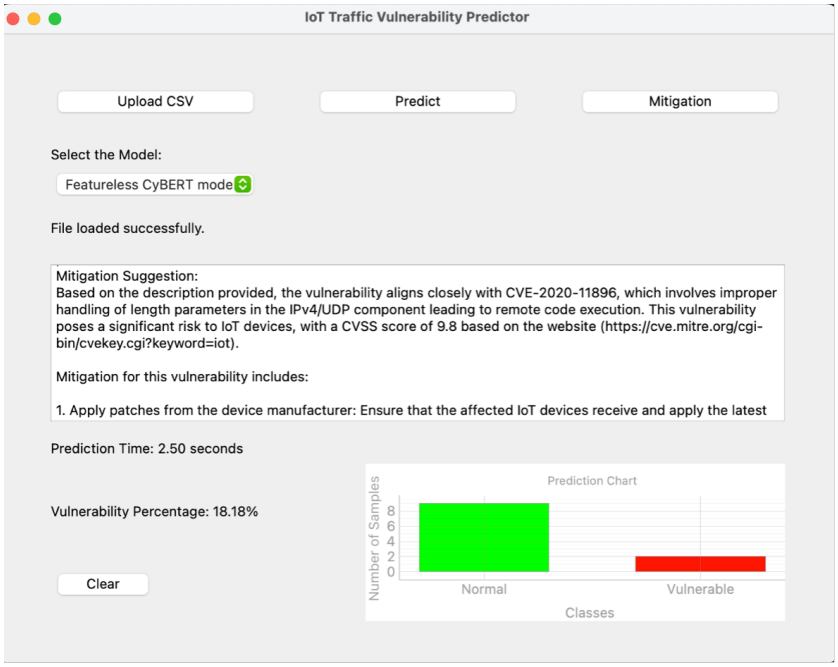

Figure 5. Featureless IoT vulnerability detection and mitigation.

5. Discussion

5.1. Contributions to the Field

The value of these study results is discussed in terms of how AI-driven DL and LLM can be effectively integrated to discover zero-day vulnerabilities across IoT networks. This work illustrates the strength of using raw network traffic data, which eliminates manual feature engineering and reduces the expertise and time required to detect vulnerabilities in IoT. Through the use of pre-trained DL and LLM, this study extends previous work in which [29] first investigated the potential of LLM in IoT security. LLMs are recursive and can be trained in any domain due to a global design pattern. We demonstrate that this approach is effective by applying it to the analysis and classification of network traffic data in cybersecurity.

Thus, with modern-day IoT security becoming increasingly important, this study aligns with the recent trend emphasizing the necessity for lightweight, accessible, and scalable tools. This study has achieved a significant milestone by developing a platform that includes a user-friendly interface and much lighter models, 277.5 MB for DL and 334 MB for LLM. This advancement makes real-world deployment on low-power and resource-constrained IoT devices feasible moving forward. Deploying lightweight and user-friendly models is crucial for IoT applications, as noted in [3,30], where computational resources are typically limited.

Unlike standard ML approaches that rely on manually built features for vulnerability identification, our methodology evaluates raw network traffic data, eliminating the need for direct feature extraction. Manual feature engineering is time-consuming and may overlook

small patterns critical for identifying emerging threats. Our featureless DL technique automatically learns these representations, enhancing detection accuracy and flexibility. Furthermore, the integration of LLM enables the production of actionable mitigation measures, resulting in an end-to-end solution for IoT security. Furthermore, the integration of LLM enables the production of actionable mitigation measures, resulting in an end-to-end solution for IoT security.

Our tests of both the Cardiff University Smart Home Lab dataset and a subset of the IoT23 dataset show that our technique outperforms traditional methods regarding processing efficiency, resilience, and overall performance. This comparison highlights the significant benefits of our methodology over previous approaches, particularly in dynamic IoT contexts where real-time detection and reaction are crucial.

What is particularly important about this study is that, compared with traditional intrusion detection systems (IDS), which often use supervised learning and feature extraction, the methodology employed here is much more featureless. Malicious software encountered by intrusion detection systems, especially zero-day vulnerabilities, can effectively identify threats by detecting unknown data patterns without relying on hard signatures from previous research [31]. Our approach of leveraging raw data to identify vulnerabilities aligns with the idea that this system should operate without, or at least minimize, frequent manual feature engineering and the unbounded adaptability of release agents against unknown threats.

The integration of LLMs into mitigating IoT vulnerabilities, particularly Ripple20 vulnerabilities, represents a significant advancement in cybersecurity. LLMs enhance the process by automating vulnerability identification through parsing network traffic and logs, enabling real-time and accurate detection. They also provide contextually tailored mitigation strategies, including specific remediation steps or configuration changes for identified CVEs. Furthermore, LLMs offer scalability by monitoring extensive IoT networks and analyzing diverse data. Their adaptability ensures that mitigation strategies remain effective against evolving threats.

5.2. Future Research Directions

This study raises several future research directions. One potential direction is to continue developing LLM architectures by optimizing them or enhancing their efficiency within IoT contexts. These models can be effectively implemented in constrained IoT devices, improving performance without compromising their lightweight nature due to hardware limitations. Moreover, it would be intriguing for these AI models to learn from new threats using adaptive mechanisms, such as reinforcement learning or continuous updates, as proposed in the work on adaptive security frameworks by [30]. Such advancements could improve detection capabilities for emerging vulnerabilities, which is in high demand given the rapidly changing cybersecurity landscape.

Integration with Explainable AI (XAI) combines LLMs with XAI techniques, providing transparent and understandable mitigation strategies to assist security professionals in their decision-making processes. This integration enhances the interpretability of LLM-driven output and addresses challenges identified in previous research.

Collaborative security frameworks that implement LLMs within collaborative platforms can facilitate the sharing of threat intelligence and mitigation strategies between organizations, thereby improving collective resilience to cybersecurity.

Concerning adversarial threats and countermeasures, our method is highly accurate in detecting IoT vulnerabilities. However, it is critical to recognize potential adversarial threats in real-world implementations. Attackers can manipulate network traffic by injecting specifically designed packets to avoid detection or confuse the system. Modifications to

packet features, such as header information, payload content, or packet timing, that seem to normalize traffic are examples of potential adversarial threats. Such manipulations can lead to false negatives, where true vulnerabilities are ignored, or false positives, where benign traffic is misidentified as malicious. Several countermeasures can enhance our detection framework, including anomaly detection and ensemble methods.

Enhanced mitigation frameworks aim to improve the mitigation process by combining expert evaluation with ensemble approaches. Domain experts will assess the developed mitigation techniques to confirm their accuracy and applicability. Concurrently, ensemble or consensus-based methodologies will be explored to aggregate findings from various models. This combined approach is expected to produce more reliable mitigation recommendations.

Lastly, introducing privacy-preserving methods such as federated learning can further enhance the separability of these models by allowing their training on data from multiple IoT network locations. However, this does not ensure that a high-accuracy detection system can maintain data privacy. One approach to address this is the development of privacy-preserving models for similar applications of IoT and IIoT devices. Partnerships between industry and academia that test these models in real-world IoT setups, as well as more diverse environments around the world, can provide valuable feedback. This feedback would help refine the design of these models, ensuring they operate robustly across all deployment scenarios.

Author Contributions: Methodology, S.B.H. and S.L.; Formal analysis, S.L.; Writing—original draft, S.B.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bizga, A. 19 Zero-Day Vulnerabilities Affect Millions of IoT Devices Worldwide. 2023. Available online: https://www.bitdefender.com/en-us/blog/hotforsecurity/19-zero-day-vulnerabilities-affect-millions-iot-devices-worldwide (accessed on 5 June 2024).
2. Statista. Number of IoT Connected Devices Worldwide 2023–2030. 2024. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed on 10 December 2024).
3. Hulayyil, S.B.; Li, S.; Xu, L. Machine-learning-based vulnerability detection and classification in internet of things device security. *Electronics* **2023**, *12*, 3927. [CrossRef]
4. Bs, S.; Nagapadma, R. P-DNN: Parallel DNN based IDS framework for the detection of IoT vulnerabilities. *Secur. Priv.* **2024**, *7*, e330.
5. Brooks, R.A. Intelligence without representation. *Artif. Intell.* **1991**, *47*, 139–159. [CrossRef]
6. Zohaib, M.; Ahsan, M.; Khan, M.; Iqbal, J. A featureless approach for object detection and tracking in dynamic environments. *PloS ONE* **2023**, *18*, e0280476.
7. Udurume, M.; Shakhov, V.; Koo, I. Comparative Analysis of Deep Convolutional Neural Network—Bidirectional Long Short-Term Memory and Machine Learning Methods in Intrusion Detection Systems. *Appl. Sci.* **2024**, *14*, 6967. [CrossRef]
8. Gueriani, A.; Kheddar, H.; Mazari, A.C. Enhancing IoT Security with CNN and LSTM-Based Intrusion Detection Systems. In Proceedings of the 2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS), El Oued, Algeria, 24–25 April 2024; IEEE: New York, NY, USA, 2024; pp. 1–7.
9. Xin, Q.; Xu, Z.; Guo, L.; Zhao, F.; Wu, B. IoT traffic classification and anomaly detection method based on deep autoencoders. *Appl. Comput. Eng.* **2024**, *69*, 64–70. [CrossRef]
10. Becerra-Suarez, F.L.; Tuesta-Monteza, V.A.; Mejia-Cabrera, H.I.; Arcila-Diaz, J. Performance Evaluation of Deep Learning Models for Classifying Cybersecurity Attacks in IoT Networks. *Informatics* **2024**, *11*, 32. [CrossRef]
11. Jones, R. Enhancing IoT Security. In *Advances in Information Security, Privacy, and Ethics Book Series*; IGI Global Scientific Publishing: Hershey, PA, USA, 2024; pp. 56–71.

12. Yosifova, V. Application of Open-source Large Language Model (LLM) for Simulation of a Vulnerable IoT System and Cybersecurity Best Practices Assistance. *Preprints* **2024**. [CrossRef]
13. Malisetty, B.; Perez, A.J. Evaluating Quantized Llama 2 Models for IoT Privacy Policy Language Generation. *Future Internet* **2024**, *16*, 224. [CrossRef]
14. Xiao, B.; Kantarci, B.; Kang, J.; Niyato, D.; Guizani, M. Efficient Prompting for LLM-based Generative Internet of Things. *arXiv* **2024**, arXiv:2406.10382. [CrossRef]
15. Egashira, K.; Vero, M.; Staab, R.; He, J.; Vechev, M. Exploiting LLM Quantization. *arXiv* **2024**, arXiv:2405.18137.
16. Refael, Y.; Hakim, A.; Greenberg, L.; Aviv, T.; Lokam, S.; Fishman, B.; Seidman, S. SLIP: Securing LLMs IP Using Weights Decomposition. *arXiv* **2024**, arXiv:2407.10886.
17. Tony, C.; Mutas, M.; Ferreyra, N.E.D.; Scandariato, R. Llmseceval: A dataset of natural language prompts for security evaluations. In Proceedings of the 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), Melbourne, VIC, Australia, 15–16 May 2023; IEEE: New York, NY, USA, 2023; pp. 588–592.
18. Saha, D.; Tarek, S.; Yahyaei, K.; Saha, S.K.; Zhou, J.; Tehranipoor, M.; Farahmandi, F. Llm for soc security: A paradigm shift. *IEEE Access* **2024**, *2*, 155498–155521. [CrossRef]
19. Imathiu, G.; Chege, A.; Omamo, A. Security intrusion monitoring model for Internet of Things (IoT) using sniffing tools on wireless sensor networks. *Afr. J. Sci. Technol. Soc. Sci.* **2023**, *2*, 51–58.
20. Garalov, T.; Elhajj, M. Enhancing IoT Security: Design and Evaluation of a Raspberry Pi-Based Intrusion Detection System. In Proceedings of the 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 23–26 October 2026; IEEE: New York, NY, USA, 2023; pp. 1–7.
21. Harwalkar, S.S.; Hussein, A.; Kumar, B.V.; Habelalmateen, M.I.; Victoria, R.M. Intrusion Detection in IoT Platform Using Tuna Swarm Optimization with Long Short-Term Memory. In Proceedings of the 2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE), Ballari, India, 2–3 November 2023; IEEE: New York, NY, USA, 2023; pp. 1–6.
22. Jafar, I.B.; Al-Anbagi, I. RSM: A Real-time Security Monitoring Platform for IoT Networks. In Proceedings of the 2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Regina, SK, Canada, 24–27 September 2023; IEEE: New York, NY, USA, 2023; pp. 393–398.
23. Khosla, K.; Jones, R.; Bowman, N. Featureless Deep Learning Methods for Automated Key-Term Extraction. 2019. Available online: https://api.semanticscholar.org/CorpusID:204805716 (accessed on 24 March 2025).
24. Pintea, S.L.; Mettes, P.S.; van Gemert, J.C.; Smeulders, A.W. Featureless: Bypassing feature extraction in action categorization. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; IEEE: New York, NY, USA, 2016; pp. 196–200.
25. Rizvi, S.M.H. Time series deep learning for robust steady-state load parameter estimation using 1D-CNN. *Arab. J. Sci. Eng.* **2022**, *47*, 2731–2744. [CrossRef]
26. Liu, L.; Si, Y.W. 1D convolutional neural networks for chart pattern classification in financial time series. *J. Supercomput.* **2022**, *78*, 14191–14214.
27. Shahid, S.M.; Ko, S.; Kwon, S. Performance Comparison of 1D and 2D Convolutional Neural Networks for Real-Time Classification of Time Series Sensor Data. In Proceedings of the 2022 International Conference on Information Networking (ICOIN), Jeju-si, Republic of Korea, 12–15 January 2022; IEEE: New York, NY, USA, 2022; pp. 507–511.
28. Szandała, T. Review and comparison of commonly used activation functions for deep neural networks. In *Bio-Inspired Neurocomputing*; Springer: Singapore, 2021; pp. 203–224.
29. Yang, Y. Iot software vulnerability detection techniques through large language model. In Proceedings of the International Conference on Formal Engineering Methods, Brisbane, QLD, Australia, 21–24 November 2023; Springer: Brisbane, QLD, Australia, 2023; pp. 285–290.
30. Ferrag, M.A.; Ndhlovu, M.; Tihanyi, N.; Cordeiro, L.C.; Debbah, M.; Lestable, T.; Thandi, N.S. Revolutionizing Cyber Threat Detection with Large Language Models: A privacy-preserving BERT-based Lightweight Model for IoT/IIoT Devices. *IEEE Access* **2024**, *12*, 23733–23750.
31. Alsoufi, M.A.; Razak, S.; Siraj, M.M.; Nafea, I.; Ghaleb, F.A.; Saeed, F.; Nasser, M. Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Appl. Sci.* **2021**, *11*, 8383. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.